LNAI 4452

Maria Fasli
Onn Shehory (Eds.)

Agent-Mediated Electronic Commerce

Automated Negotiation and Strategy Design for Electronic Markets

AAMAS 2006 Workshop, TADA/AMEC 2006
Hakodate, Japan, May 2006
Selected and Revised Papers

Springer

Maria Fasli   Onn Shehory (Eds.)

Agent-Mediated Electronic Commerce

Automated Negotiation and Strategy Design for Electronic Markets

AAMAS 2006 Workshop, TADA/AMEC 2006
Hakodate, Japan, May 9, 2006
Selected and Revised Papers

Springer

Series Editors

Jaime G. Carbonell, Carnegie Mellon University, Pittsburgh, PA, USA
Jörg Siekmann, University of Saarland, Saarbrücken, Germany

Volume Editors

Maria Fasli
University of Essex, Department of Computer Science
Wivenhoe Park, Colchester CO4 3SQ, UK
E-mail: mfasli@essex.ac.uk

Onn Shehory
IBM Haifa Research Lab, Tel Aviv Site
Haifa University Campus, Mount Carmel, Haifa 31905, Israel
E-mail: onn@il.ibm.com

Library of Congress Control Number: 2007926025

CR Subject Classification (1998): I.2.11, K.4.4, C.2, H.3.4-5, H.5.3, I.2, J.1

LNCS Sublibrary: SL 7 – Artificial Intelligence

ISSN	0302-9743
ISBN-10	3-540-72501-6 Springer Berlin Heidelberg New York
ISBN-13	978-3-540-72501-5 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper   SPIN: 12063871   06/3180   5 4 3 2 1 0

Preface

The design and analysis of trading agents and electronic trading systems in which they are deployed involve finding solutions to a diverse set of problems, involving individual behaviors, interaction, and collective behavior in the context of trade. A wide variety of trading scenarios and systems, and agent approaches to these, have been studied in recent years. The present volume includes a number of papers that were presented as part of the Joint International Workshop on Trading Agent Design and Analysis and Agent-Mediated Electronic Commerce which was collocated with the Autonomous Agents and Multi-agent Systems (AAMAS) Conference in Hakodate, Japan, in May 2006.

The Joint TADA/AMEC Workshop brought together the two successful and well-established events of the Trading Agent Design and Analysis (TADA) and Agent-Mediated Electronic Commerce (AMEC) Workshops. The TADA series of workshops serves as a forum for presenting work on trading agent design and technologies, theoretical and empirical evaluation of strategies in complex trading scenarios as well as mechanism design. TADA also serves as the main forum for the Trading Agent Competition (TAC) research community. TAC is an annual tournament whose purpose is to stimulate research in trading agents and market mechanisms by providing a platform for agents competing in well-defined market scenarios (http://www.sics.se/tac). The AMEC series of workshops presents interdisciplinary research on both theoretical and practical issues of agent-mediated electronic commerce ranging from the design of electronic marketplaces and efficient protocols to behavioral aspects of agents operating in such environments. The merging of the two workshops was a unique opportunity for researchers working in agents and multi-agent systems, artificial intelligence, operational research, economics and game theory to explore issues pertinent to the development of agent-populated electronic markets. The collection of papers in this volume provides a glimpse into this wide field of research.

The papers presented at the workshop contribute to the theory and practice of agent-based electronic trade and commerce addressing both the agent level and the system level. The papers presented included work directly related to TAC, work related to generic markets and trading scenarios, theoretical and experimental studies, automated negotiation, market mechanism design as well as strategy design.

We hope that this collection of papers will be a useful resource for researchers, practitioners and students working in automated trading and electronic marketplaces.

March 2007

Maria Fasli
Onn Shehory

Organization

Workshop Chairs

Maria Fasli, University of Essex, UK (TADA Chair)
Onn Shehory, IBM - Haifa Research Lab, Israel (AMEC Chair)

Program Committee

Chris Brooks, University of San Francisco, USA
Dave Cliff, University of Southampton/Syritta Algorithmics, UK
John Collins, University of Minnesota, USA
Marc Esteva, IIIA-CSIC, Spain
Boi Faltings, EPFL, Switzerland
Shaheen Fatima, Liverpool University, UK
Amy Greenwald, Brown University, USA
Minhua He, University of Southampton, UK
Sverker Janson, SICS, Sweden
Wolf Ketter, University of Minnesota, USA
Kate Larson, University of Waterloo, Canada
Kevin Leyton-Brown, University of British Columbia, Canada
Tracy Mullen, Penn State University, USA
Julian Padget, University of Bath, UK
David Pardoe, University of Texas at Austin, USA
Sun Park, Yonsei University, Korea
Simon Parsons, Brooklyn College, USA
Jeff Rosenschein, Hebrew University, Israel
Norman Sadeh, Carnegie Mellon University, USA
Alberto Sardinha, ScD - Computer Science - PUC-Rio, Brazil
Carles Sierra, IIIA, Spain
Peter Stone, University of Texas at Austin, USA
Ioannis Vetsikas, University of Southampton, UK
Michael Wellman, University of Michigan, USA
Steven Willmott, UPC, Spain
Dongmo Zhang, University of Western Sydney, Australia

Additional Reviewers

Viswanath Avasarala, Penn State University, USA
Peter Gradwell, University of Bath, UK
Terence Kelly, Hewlett-Packard Labs Palo Alto, USA
Christopher Kiekintveld, University of Michigan, USA
Xudong Luo, University of Southampton, UK
Casey Marks, Brown University, USA
Lars Rasmusson, SICS, Sweden

Table of Contents

Evolutionary Optimization of ZIP60: A Controlled Explosion in Hyperspace

Dave Cliff

Foreign Exchange Complex Risk Group, Deutsche Bank
1 Great Winchester Street, London EC2N 2DB
dave.cliff@bcs.org

Abstract. The "ZIP" adaptive trading algorithm has been demonstrated to outperform human traders in experimental studies of continuous double auction (CDA) markets. The original ZIP algorithm requires the values of eight control parameters to be set correctly. A new extension of the ZIP algorithm, called ZIP60, requires the values of 60 parameters to be set correctly. ZIP60 is shown here to produce significantly better results than the original ZIP (called "ZIP8" hereafter), for negligable additional computational costs. A genetic algorithm (GA) is used to search the 60-dimensional ZIP60 parameter space, and it finds parameter vectors that yield ZIP60 traders with mean scores significantly better than those of ZIP8s. This paper shows that the optimizing evolutionary search works best when the GA itself controls the dimensionality of the search-space, so that the search commences in an 8-d space and thereafter the dimensionality of the search-space is gradually increased by the GA until it is exploring a 60-d space. Furthermore, the results from ZIP60 cast some doubt on prior ZIP8 results concerning the evolution of new 'hybrid' auction mechanisms that appeared to be better than the CDA.

1 Introduction

The Zero-Intelligence Plus (ZIP) adaptive automated trading algorithm [6] has been demonstrated to outperform human traders in experimental studies of continuous double auction (CDA) markets populated by mixtures of human and "robot" traders [15]. To successfully populate a market with ZIP traders, the values of eight real-valued control parameters need to be set correctly. While these eight values can of course be set manually, previous papers have demonstrated that this 8-d parameter-value vector can be automatically optimized using a simple genetic algorithm (GA) search to tailor ZIP traders to particular markets, thereby producing results superior to those from ZIP traders with manually-set parameter values [7, 8]. Furthermore, a simple extension of the GA-ZIP approach (i.e., adding a single additional real-valued numeric parameter, its value set by the GA) allows for automated market-mechanism design, and has been demonstrated as a possible way of automatically discovering novel "hybrid" forms of auction mechanism that appear to be more efficient than the CDA [10, 11, 12]. This paper introduces a more sophisticated version of the ZIP algorithm, which is shown to produce significantly better results. The extended variant is known as "ZIP60", because it requires 60 real-valued control parameters to be set correctly, and thus the original algorithm is now re-named as "ZIP8". Manually identifying the correct values for 60 control parameters could be a very laborious task, but it is demonstrated here that an appropriate automatic search or optimization process (such as a GA) can reliably discover good sets of values for the parameters, so long as some care is exercised in controlling a gradual expansion of the dimensionality of the search-space. The GA operating in the 60-dimensional parameter space is shown to produce markets populated by ZIP60 traders with mean scores significantly better than those of ZIP8s. Moreover, the ZIP60 results presented in this paper, while better than ZIP8, show a markedly reduced incidence of cases where the GA also discovers novel hybrid auction mechanisms within which the ZIP traders perform significantly better than when they interact within the fixed CDA mechanism. A plausible conclusion drawn from this is that it indicates that the earlier ZIP8 results (where apparent "improvements" on the CDA were common) were actually consequences of the relative lack of sophistication in the ZIP8 algorithm, rather than consequences of previously-undiscovered weaknesses in the CDA mechanism that the ZIP8 traders were operating within.

In the interests of scientific openness and ease of replicability, the C source-code that was used to generate the ZIP60 results in this paper has been published in a technical report freely available on the web [13].

This paper reports on an ongoing line of research, and there are several open avenues of research that could be pursued to extend or further explore the ideas presented here. In particular, it is important to note that the results in this paper are certainly not intended as an absolute and conclusive demonstration that ZIP60 is superior to all other CDA bidding algorithms, or that the solutions discovered by the GA are optimal in the sense of the GA routinely discovering Nash equilibria in the experimental markets that ZIP60 is studied within here. This paper studies the equilibrating performance of markets that are homogeneously populated with one type of trader-agent, in the style of frequently-cited prior work such as that by Gode & Sunder [20], Cliff [6, 9, 12], Preist & van Tol [30], and Gjerstad & Dickhaut [19]; rather than studying strategic interactions within markets heterogeneously populated by two or more different types of trading algorithms or market mechanisms, such as is exemplified by [38, 39, & 29]. Although the original paper [6] that introduced the ZIP8 algorithm also studied ZIP8's performance *only* in homogeneously populated markets, nevertheless ZIP8 was subsequently used as a benchmark trading algorithm in numerous studies of strategic interactions between heterogeneous mixes of trading algorithms, performed by several independent groups of researchers. The number of such papers in which ZIP8 (or close derivatives of ZIP8) have been used is fairly large, and the list includes: [15, 38, 39, 23, 29, 40, & 1]. Thus, given that so much prior work exploring strategic interactions and heterogeneous populations has been based on ZIP8, it seems reasonable at least to presume that researchers with an interest in studying heterogeneous marketplaces might find ZIP60 a useful new benchmark, even though this current paper reports only on ZIP60 in homogeneous settings. While the study of ZIP60's strategic interactions with other CDA bidding algorithms is certainly an important topic of further research, it is beyond the scope of this current paper.

Furthermore, it is worth noting that in pretty much all of the above-cited papers studying strategic interactions between heterogeneous mixtures of bidding algorithms, the results come from experiments in which the nature of the market supply and demand curves are essentially fixed for the duration of each experiment. That is, studies exploring the effects that significant changes to the supply or demand (or both) curves can have on the trading-agent market's internal dynamics seem pretty rare. Most often, the supply and demand curves in any one trader-agent experiment remain largely the same for the entire duration of that experiment. This seems very curious, given that one commonly-claimed motivation for studying market systems is that mechanisms such as the CDA are interesting because of their ability to quickly and robustly adapt to dynamic and unexpected changes in supply and/or demand; that studies of shock-changes in *human* CDA markets date back as far as Vernon Smith's seminal 1962 paper [36]; and that such changes are known to occur in real-world markets.[1] If CDA markets are interesting because they exhibit attractive adaptation to dynamic changes in supply and demand, why is there this affection in the trading-agent literature for studying CDA systems where such changes are largely absent? In contrast, the results reported in this paper all come from experiments in which the marketplaces periodically undergo sudden "shock" changes to the supply and/or demand curves, and where the ZIP60 traders are optimized on the basis of their ability to rapidly and stably adapt to the new market conditions prevailing after each shock-change.

The rest of this paper is structured as follows. Section 2 gives an overview of ZIP traders and of the experimental methods used, including a description of the continuously variable space of auction types. This description is largely identical to the account given in previous papers (e.g., [10, 12]), albeit extended to describe how the new experiments whose results are presented here differ from the previous work. The new ZIP60 results are then presented, analyzed, and discussed in Section 3.

2 Methods

2.1 The Original Eight-Parameter ZIP

The original eight-parameter ZIP trading algorithm was first described fully in a lengthy report [6], which included source-code (in ANSI C) of an example implementation. For the purposes of this paper, a high-level description of the algorithm and its eight key parameters is sufficient. Illustrative C source-code for ZIP60 has been published in [13]. As will be seen in Section 3, there are in fact a family of ZIP algorithms between ZIP8 and ZIP60, and so hereinafter the acronym "ZIP" with no numeric suffix is intended to mean "all ZIP*n* for $8 \le n \le 60$ and beyond".

ZIP traders deal in arbitrary abstract commodities. Each ZIP trader *i* is given a private (i.e., secret) limit-price, λ_i, which for a seller is the price below which it must not sell and for a buyer is the price above which it must not buy. If a ZIP trader completes a transaction at its λ_i price then it generates zero utility ("profit" for the sellers or "saving" for the buyers). For this reason, each ZIP trader *i* maintains a time-varying utility margin $\mu_i(t)$ and generates quote-prices $p_i(t)$ at time *t* using $p_i(t)=\lambda_i(1+\mu_i(t))$ for sellers and $p_i(t)=\lambda_i(1-\mu_i(t))$ for buyers. The "aim" of traders is to

[1] E.g., in high-frequency foreign-exchange price time series, "gap" step-changes in price are not unusual.

maximize their utility over all trades, where utility is the difference between the accepted quote-price and the trader's λ_i value. Trader i is given an initial value $\mu_i(0)$ (i.e., $\mu_i(t)$ for $t=0$) which is subsequently adapted over time using a simple machine learning technique known as *the Widrow-Hoff rule* which is also used in back-propagation neural networks and in learning classifier systems. This rule has a "learning rate" parameter β_i that governs the speed of convergence between trader i's quoted price $p_i(t)$ and the trader's idealized "target" price $\tau_i(t)$. When calculating $\tau_i(t)$, ZIP traders introduce a small random *absolute* perturbation generated from[2] $\boldsymbol{U}[0,c_a]$ (this perturbation is positive when increasing $\tau_i(t)$, negative when decreasing) and also a small random *relative* perturbation generated from $\boldsymbol{U}[1\text{-}c_r,1]$ when decreasing $\tau_i(t)$, or from $\boldsymbol{U}[1,1+c_r]$ when increasing $\tau_i(t)$, where c_a and c_r are global system constants. To smooth over noise in the learning system, there is an additional "momentum" parameter γ_i for each trader (such momentum terms are also common in back-propagation neural networks).

So, adaptation in each ZIP trader i has the following parameters: initial margin $\mu_i(0)$; learning rate β_i; and momentum term γ_i. In an entire market populated by ZIP traders, values for these three parameters are randomly assigned to each trader via $\mu_i(0)=f_a(\mu_{min}, \mu_\Delta)$, $\beta_i=f_a(\beta_{min}, \beta_\Delta)$, and $\gamma_i=f_a(\gamma_{min}, \gamma_\Delta)$; for $f_a(\alpha, \kappa)=\boldsymbol{U}[\alpha, \alpha+\kappa]$. Hence, to initialize an entire ZIP-trader market, it is necessary to specify values for the six market-initialization parameters μ_{min}, μ_Δ, β_{min}, β_Δ, γ_{min}, and γ_Δ; and for the two system constants c_a and c_r. Thus any set of initialization parameters for a ZIP-trader market exists within an eight-dimensional real space – hence "ZIP8".

Vectors in this 8-space can be considered as "genotypes" in a genetic algorithm (GA), and from an initial population of randomly generated genotypes it is possible to allow a GA to find new genotype vectors that best satisfy an appropriate evaluation function. This is exactly the process that was first introduced in [7, 8]. For the purposes of this paper, we will consider the GA optimizer as a "black box" and leave it largely un-discussed: full details accompany the source-code in [13].

In addition to using the GA to optimize the control parameters for the trader-agents, one more real-valued numeric parameter was introduced in [10–12] to give the GA automated control over the auction mechanism. This market-mechanism parameter is called Q_s and it governs the *exogenously imposed* probability that the next quote in the marketplace will be taken from a seller, so Q_s=0.0 is a pure one-sided auction where only buyers can quote (and hence is similar to an English auction); Q_s=1.0 is pure one-sided with only sellers quoting (as in a Dutch Flower auction); and Q_s=0.5 makes quotes from buyers or sellers equi-probable (as in a CDA). The surprising result reported in [10–12] is that "hybrid" auction mechanisms (such as Q_s=0.25) were found by the GA to give the best evaluation scores when the value of Q_s was evolved alongside the values of the eight ZIP control parameters. Experiments where the value of Q_s was under control of the GA are referred to here as "EM" (for "evolving mechanism") experiments, and experiments where the value of

[2] Here $v=\boldsymbol{U}[x,y]$ denotes a random real value v generated from a uniform distribution over the range $[x,y]$.

Q_s was fixed, typically at the CDA value of 0.5, are referred to as "FM" experiments (for "fixed mechanism").

The fitness of genotypes was evaluated here using the methods described previously [7, 8, 10–12]: one *trial* of a particular genome was performed by initializing a ZIP-trader market from the genome, and then allowing the ZIP traders to operate within the market for a fixed number of trading periods (often colloquially referred to as "days"), with allocations of stock and currency being replenished between each trading period. During each trading period, Smith's [36] α measure (root mean square deviation of transaction prices from the market's theoretical competitive equilibrium price) was monitored, and a weighted average of α was calculated across the days in the trial, using a method described in more detail in the next section. As the outcome of any one such trial is influenced by stochasticity in the system, the final evaluation score for an individual was calculated as the arithmetic mean of 100 such trials. Note that as *minimal* deviation of transaction prices from the theoretical equilibrium price is desirable, lower scores are better: we aim here to *minimize* the evaluation scores. That is, individuals with lower scores have greater reproductive fitness.

2.2 Previous ZIP8 Results

In [12], results from 32 sets of experiments were published, where each experiment involved sequences built from one or more of four specific market supply and demand schedules. These four schedules are referred to as markets M1, M2, M3, and M4, and are illustrated in [12, 13]. In all four schedules there are 11 buyers and 11 sellers, each empowered to buy/sell one unit of commodity. Market M1 is taken from Smith's seminal 1962 paper [36] on his early experimental economics work, and the remaining three markets are variations on M1. In M2 the slope of the demand curve has been greatly reduced while the slope of the supply curve has been increased only slightly; and in M4 the slope of the supply curve has been greatly reduced while the slope of the demand curve has been increased only slightly. In M3 the slopes of both the supply and demand curves are only slightly steeper than the slopes in M1, yet these minor differences between the supply and demand curves in M1 and M3 can still lead to significant differences in the final best evolved solutions.

The experiments reported in this paper use a method first explored in ZIP8 experiments, involving "shock changes" being inflicted on the market by swapping from one schedule to another partway through the evaluation process. Here, two shocks occurred during each evaluation process (i.e., switching between three schedules). For instance, in one experiment referred to here as M121, the evaluation involved six trading periods ("days") with supply and demand determined by M1, then a sudden change to M2, then six periods/days later a reversion to M1 for a final six periods. The other sets of experiments are similarly named M212, M123, M321, and so on. Each of the three market schedules was used for six "days", so the two-shock trials last for 18 days. As in the previous GA-ZIP work, the evaluation function was a weighted average of Smith's [36] "α" measure of root mean square deviation of transaction prices from the underlying theoretical equilibrium price at the start of the experiment, measured across the six periods for each schedule used: in each trading period p the value α_p was calculated, and the evaluation score was computed as

$(1/\Sigma w_p).\Sigma(\alpha_p.w_p)$ for p=1…18 with weights w_1=1.75, w_2=1.5, w_3=1.25, $w_{3<p<7}$=1.0, $w_{p>6}=w_{p-6}$, and $w_{p>12}=w_{p-12}$.

The process used to compare the EM and FM cases is as follows. In any one experiment, here involving a population of 30 genotypes over 500 generations, in each generation the elite (best-scoring) individual is of most interest, and so the time-series of the elite fitness score for the population is monitored across the 500 generations. These results are non-deterministic: different runs of the GA (with different seed values for its random number generator) will yield different elite trajectories. Examining the results from 50 repetitions of an experiment (varying only the random seed between repetitions) often gives multimodal results, and in all experiments we are interested only in the best elite mode (i.e. the mode with lowest scores), which can be summarized by the mean and standard deviation (s.d.) of the scores within that mode at each generation: these two values will be referred to here as the *best elite-mode* fitness mean and s.d.. For comparison purposes, in the ZIP8 work reported in [12], similar trajectories of best elite-mode fitness values were recorded from 50 repetitions of the each experiment in fixed-mechanism (FM) conditions, where the value of Q_s was *not* evolved but instead was fixed at the CDA value of Q_s=0.5.

The results from 18 dual-shock (triple-schedule) experiments were presented in four separate data-tables in [12], grouped by the nature of the shocks (i.e., the "treatment regime"). Table 3 showed results from experiments where only the demand curve undergoes a major change on each shock (i.e.: M121, M212, M232, M323, M123, and M321). Table 4 showed results from experiments where only the supply curve undergoes a major change on each shock (i.e.: M141, M414, M434, M343, M143, and M341). In Table 5, one of the two shocks involves a major change only to the demand curve while the other shock involves a major change only to the supply curve (i.e.: M432, M234, M412, and M214); and in Table 6 each shock involved a major change to both the supply curve and the demand curve (i.e.: M242 and M424). In this paper, all 18 dual-shock results are shown together in a single graph, but the results appear in table order, as was just listed.

Analysis of the ZIP8 results showed that the GA never failed to discover EM genotypes that were at least as good (i.e. had elite evaluation scores at least as low) as the corresponding FM genotypes, and in several cases the EM result was significantly better (lower) than the FM result, at the 1% confidence level, using appropriate non-parametric significance tests such as the Wilcoxon-Mann-Whitney (see, e.g., [35]), or latterly the Robust Rank Order test [16].

The histogram in Figure 1 shows the results for GA-optimized ZIP8 in FM and EM conditions. Fig.1 also shows the results from various styles of ZIP60 EM experiments, discussed further in Section 3 of this paper. The ZIP8 statistics in Fig.1 are the results of conducting a more rigorous and careful analysis (discussed in [13]) of the data than was originally summarized and tabulated in [12]. The final evaluation score recorded as the outcome of any one experiment is now taken as an average of the final few elite scores (over generations 490 to 500) to smooth over noise in the evaluation process; and the summary statistics for each type of experiment are here always calculated from the top 10% (i.e., the upper decile) of the 50 repetitions of each type of experiment, regardless of how many repetitions converged on solutions with final elite scores in the best elite mode. So, the data in Fig.1 show the mean and s.d. of the final outcome elite scores from the best (lowest-scoring) five experiments in each study.

2.3 Related Work

These previous GA-ZIP results have subsequently been replicated, adapted, and extended in a number of independent studies. Robinson [32] explored the use of evolved market-mechanisms in the context of market-based control (e.g. [4]) of scarce resources in utility-scale corporate data centers. Walia [41] explored the use of the same evolving-mechanism techniques but with markets populated by Gode & Sunder's [20] ZI trader-agents rather than ZIP traders, again finding evidence that non-standard hybrid mechanisms were discovered as good/best solutions by the GA; and Byde [2] demonstrated that the same techniques could lead to the evolution of hybrid sealed-bid auction mechanisms, regardless of the type of trader operating in the market. Shipp [34] investigated how the nature of the evolved solutions changed as the number of "market shocks" used in the evaluation process increased; and Wichett [43] explored a system in which multiple reproductively separate "gene-pools" of ZIP traders competed, co-adapted, and co-evolved along with the market mechanism. Other recent uses of ZIP include modifying it for bargaining in sealed-bid auctions [1]; using ZIP traders to study speculative trading in business-to-business exchanges [25]; and using ZIP traders to explore issues of reputation and information quality in a variety of market configurations [24].

The results in [10] were the first demonstration that radically new market mechanisms for artificial traders may be designed by automatic means. But, at much the same time as they were being generated, Steve Phelps and his colleagues were independently working on a conceptually very similar (but algorithmically rather different) theme of using artificial evolution to develop and study new auction-market mechanisms [29]. In addition to the contemporaneous work of Phelps *et al.,* a number of other authors have more recently reported on the results of using artificial evolution and other forms of automated search, learning, or optimization for exploring spaces of possible trader-agent strategies, and possible new auction mechanisms, generally with positive results [39, 18, 26, 28, 21, 27, 31, & 42]. Of course, the paper introducing ZIP [6] was not the first-ever study of artificial trading agents in double-auction markets; notable prior work includes [44], [17], and [33]. Also, [19] was developed independently at much the same time. For additional discussion of earlier work, see [6].

3 ZIP60

3.1 From 8 to 60 in Five Paragraphs

The results from using a GA to fine-tune the ZIP8 trader were sufficiently encouraging that they provoke the question of whether new variants of ZIP can be developed to take advantage of the fact that we can now (generally, at least) rely on automated optimizers like the GA to set appropriate values for the numeric parameters affecting the traders. If we commit to using an optimizer to set the parameter values, we don't need to keep the number of parameters small enough for them all to be manageable or comprehensible by humans. That's the rationale for ZIP60.

To this end, observe that in ZIP8 the genome specifies the same vector of eight real values $\{\mu_{min}, \mu_{\Delta}, \beta_{min}, \beta_{\Delta}, \gamma_{min}, \gamma_{\Delta}, c_a, c_r\}$ whether the trader is a buyer or a seller. But in some situations it's plausible that the market dynamics might be better if the parameter-values used by the buyers were different to those used by the sellers, so we could in principle have a GA-ZIP system dealing with these two cases (i.e. where *Case 1* is that the trader is a buyer; *Case 2* is that the trader is a seller) and hence optimizing sixteen real parameters (i.e., "ZIP16"), with the first vector of eight values being used to initialize the buyers and the second being used to initialize the sellers.

Next, note that in some situations a ZIP trader (whether it is a buyer or a seller) has to increase its margin, and in others it has to decrease its margin, and that it may be useful to have different parameter-values depending on which of the four cases we are in, i.e. whether the trader is a buyer raising its margin, a buyer lowering its margin, a seller raising, or a seller lowering. That's 4 cases, each with 8 values, and so "ZIP32". But we can then additionally note that, in the original specification of the ZIP algorithm, both for buyers and for sellers, there are actually *three* different cases or circumstances in which the trader alters its margin (see [6] pp.42-43 for the details of and rationale for this design). For example, a seller's margin is *raised* if *one* condition holds true (i.e., if the last quote was accepted and the seller's current price is less than the price of the current quote); but a seller's margin is *lowered* if either of *two* other possible conditions are true (i.e.: if the last quote was an accepted bid and the seller is active and the seller's price is greater than the price of the last quote; *or* if the last quote was an offer that was accepted and the seller is active and its price is greater than the price of the last quote). So we could have the genome specify *three* corresponding parameter-value vectors for the buyers and also *three* such vectors for the sellers, i.e. a total of six different vectors for six different cases, which at eight values per vector gives us "ZIP48".

And in a final flourish of parameter-count inflation, let's abandon the use of a mere pair of system-wide global constants c_a and c_r and in place initialize each trader i with its own corresponding "personal" values $c_{a,i}$ and $c_{r,i,}$ generated at initialization from the uniform distributions $\boldsymbol{U}[c_{a:min}, c_{a:min}+c_{a:\Delta}]$ & $\boldsymbol{U}[c_{r:min}, c_{r:min}+c_{r:\Delta}]$. This addition of extra parameters still allows solutions involving the old system-wide constant c_a and c_r values to be "discovered" by the GA – that will happen if better evaluation scores are associated with (near-)zero values of $c_{a:\Delta}$ and $c_{r:\Delta}$. So, the parameter-value vectors for each case needs now to specify not only the six previous system parameters (μ_{min}, μ_{Δ}, β_{min}, β_{Δ}, γ_{min}, and γ_{Δ}) but also the values for the four newly-introduced system parameters $c_{a:min}$, $c_{a:\Delta}$, $c_{r:min}$, and $c_{r:\Delta}$ – i.e., ten values per vector. For six cases, each with ten values per vector, we get to sixty values: "ZIP60".

It is worth noting that this final increase from eight to ten parameter-values per case could also be applied to any of the other ZIP*n* versions mentioned in the preceding paragraphs. That is, by the expansion of the specification of c_a and c_r, ZIP8 becomes ZIP10; ZIP16 becomes ZIP20; and ZIP32 becomes ZIP40.

We need also to introduce some terminology that will ease the analysis and discussion that come later. While a ZIP8 trader has one genetically-specified value for each parameter (so, for example, it has only one β_{min} value), a ZIP60 genome specifies six related parameter values – one for each case – which we will refer to by adding case-numbers to the subscript (e.g.: $\beta_{min:1}$, $\beta_{min:2}$, ..., $\beta_{min:6}$). For ZIP60, the entire set of sixty parameters can be generated from the pattern $P_{t:n}$ where P is one of $\{\mu, \beta, \gamma, c_a, c_r\}$; t is one of $\{min, \Delta\}$; and n is an integer in $\{1,...,6\}$. We'll refer to the set of six values for any one parameter-type (i.e., $\{P_{t:1}, P_{t:2}, ..., P_{t:6}\}$ for some given P and t) as the *homologous set* of P_t parameter values.

Finally, note that the additional computational costs of using ZIP60 as a replacement for ZIP8 are virtually zero. The space costs are those incurred in storing the additional 52 real-valued parameters: this is a large percentage increase, but in absolute terms it is still a very small amount of storage when expressed as actual additional bytes required. The additional time costs are also very low indeed: a tiny amount of extra processing is needed in initialising the ZIP60 trader (i.e., populating its look-up table of 60 real values) and then in doing table look-up while the trader is operating (i.e. choosing the values to use that are appropriate to the current "case"), but that's it.

3.2 ZIP60 Results: Control of Search-Space Dimensionality Required

In testing the performance of ZIP60, all effort thus far has been devoted to exploring the performance of ZIP60 on dual-shock tests: if markets populated by ZIP traders cannot cope with sudden shock-changes in supply and demand, then they are of little interest. Moreover, it seems highly likely (but has not yet actually been empirically verified) that if ZIP60 does better than ZIP8 on these multi-shock tests, then it will also do better in those cases where there are fewer or no market shocks.

Experience with GA optimisation of ZIP60 indicates that significant care is needed in managing the dimensionality of the search-space: simply applying the old methods that worked well with ZIP8 does *not* give best results when working with ZIP60. This is a lesson learnt from experience: for the very first attempts at evolutionary optimization of ZIP60 traders, the same experiment methods as described in Section 2 were used, except that the initial population was composed entirely of randomly generated ZIP60 individuals, rather than ZIP8s. The results from these attempts were somewhat mixed. Although the scores of the elite evolved ZIP60 traders were on average significantly better than the elite ZIP8 scores in the same experiments, the standard deviation on that average improvement was almost identical to the mean improvement itself. This large standard deviation was a reflection of the fact that, in a few cases, the evolved elite ZIP60 results were actually significantly *worse* than the corresponding ZIP8 results. Now there is nothing preventing the ZIP60 GA system from evolving genotypes that correspond to ZIP8 solutions, so it seems peculiar that the ZIP60s perform worse than the ZIP8s in some cases. There are certainly points within the ZIP60 genome-space that correspond perfectly to ZIP8 solutions: if for each of the ten homologous sets the within-set variance of the parameter values for the set is (near) zero, then that ZIP60 genome is functionally equivalent to the corresponding single-case ZIP10 genome; and furthermore if the values of the $c_{a:\Delta}$ and $c_{r:\Delta}$ homologous sets are all zero, then the ZIP60 is functioning as a ZIP8. So, how come the ZIP60 results are sometimes worse than ZIP8? The fact that the GA failed to find ZIP8 solutions within the ZIP60 genome space is a strong indication that the 60-dimensional search space has characteristics (such as local maxima, sharp ridges, and plateaus in the fitness landscape) which make the search for good genomes a nontrivial process.

To address this, the ZIP genetic encoding was extended, allowing the number of cases (1, 2, 4, or 6, as discussed in Section 3.1) to be specified *on the genome itself.* The rest of the genome is still a set of ten homologous-set vectors (each made of six real numbers). If an individual's gene specifying the number of cases is set to one, then all six parameter-values are set to be identical within each homologous set, by copying the values from the first element of the set into the remaining five. If the number of cases is set to two, then the three buyer-case parameter values within each set are forced to be identical copies of each other, as are the three seller values; and if the number of cases is set to be six, then the three buyer and the three seller parameters can all be different numeric values. Thus, the ZIP60 genomes are always 60 parameter-values long, but over-writing duplication of values within the genome can reduce the *effective* dimensionality of the parameter-vectors encoded on a particular genome so that it codes for *any* of the family of ZIP algorithms between ZIP60 and ZIP8.

The motivating hypothesis for placing the dimensionality of the search-space under evolutionary control was the belief that the GA's evolutionary search would be more successful if it could start by first simply optimizing the 1-case genome, and then (only once all the values are approximately correct) could successive multi-case refinements be progressively introduced by the GA as necessary. So, for example, if a 1-case individual mutated to become a high-case individual, thereby decoupling its genome-values across the different cases, such a mutant would only be retained in the population if the mutation that increases the number of cases is also associated with higher fitness. Strictly speaking, the initial case-increasing mutation is selectively neutral: the genome values for the different cases start out as identical copies of each other, but the case-increasing mutation allows *subsequent* mutations to introduce differences across cases, and it is those mutations that will be retained if they are correlated with higher fitness. Handing evolutionary control of the dimensionality of a search-space to the GA that is searching that space is an idea that was first explored in depth in Harvey's [22] thesis, where he developed the "species adaptation genetic algorithm", which was first successfully applied in evolving neural-network controllers for autonomous mobile physical robots [5].[3]

Two new sets of ZIP60 experiments were performed to test the effects of GA-controlled dimensionality. In the first set, the population was initialised with individuals that had a randomly-assigned value for the number of cases on their genome, with the values 1, 2, 4, and 6 being equally probable. This is the initialization we refer to here as ZIP60(1:6) (for "from 1 case to 6 cases"). In the second set, every individual in the initial population was set to have a 1-case genome; this is referred to here as the ZIP60(1:1) initialization. And so the first set of experiments, where all individuals in the initial populations were 6-case individuals, are referred to as ZIP60(6:6). Results from the EM experiments with ZIP60 with the (1:1), (1:6), and (6:6) initializations are shown in Figure 1, with ZIP8 EM and FM scores shown alongside, for comparison.

The histogram in Figure 1 shows the mean elite ZIP60 EM scores alongside the ZIP8 EM and FM scores: ZIP60 consistently out-performs ZIP8, and the error bars

[3] A recent paper by Stanley & Miikkulainen [37] re-discovers some of Harvey's [22] ideas of evolutionarily controlled dimensionality increase, which Stanley & Miikkulainen rename as "complexification".

showing the s.d. values make it clear that these differences are significant. On the average, the ZIP60(1:1) scores are 14.0% better (lower) than the ZIP8 scores (and the s.d. on that mean improvement is 5.7%). In comparison, the ZIP60(6:6) scores are on average 12.91% better than the ZIP8, but the s.d. on that improvement is 12.88%; and for ZIP60(1:6), the average improvement is 12.32% with s.d.=7.03%. So, ZIP60(1:1) has the highest mean increase in performance and the lowest s.d. on its mean increase.

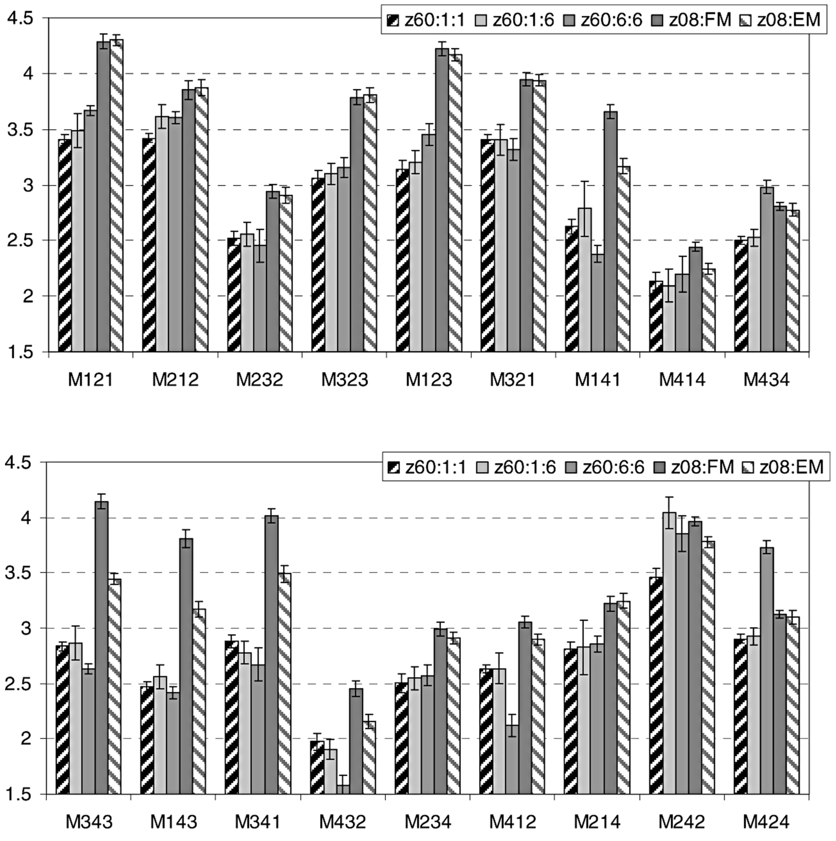


Fig. 1. Mean elite outcome scores from the best 10% (n=5) of the 50 repetitions of each of the 18 "dual-shock" experiments involving two sudden changes in market supply and demand function, as described in the text. Labels on the horizontal axis indicate the specific shock sequence. Vertical axis is evaluation score: a weighted average of root mean square deviation of transaction prices from the theoretical competitive equilibrium price, expressed as a percentage of the equilibrium price; a metric labeled "α" by Smith [36]. Lower scores are better. Each bar in the graph shows a mean score, with error bars at plus and minus one s.d. For each shock-sequence, the cluster of 5 bars shows the results for (from left to right): EM ZIP60(1:1); EM ZIP60(1:6); EM ZIP60(6:6); FM ZIP8, & EM ZIP8. See text for further discussion.

Results from significance analysis of the differences between the ZIP60(1:1) and ZIP60(1:6) upper-decile elite scores for the 18 dual-shock experiment schedules are tabulated in [13], and they offer weakly supportive evidence for the claim that ZIP60(1:1) is a better initialization than ZIP60(1:6). Using the Robust Rank Order test [16] at the 1% significance level reveals that, over the 18 types of experiment, only for schedule M242 does ZIP60(1:1) lead to significantly better results than ZIP60(1:6). In all other cases, no statistically significant difference in the scores is detected. So, ZIP60(1:1) is certainly no worse than ZIP60(1:6), and the evidence thus far is that it is actually significantly better in one of the 18 cases studied. The absence of a huge difference is perhaps no surprise given that a ZIP60(1:1) system will, after sufficiently many generations, be pretty much indistinguishable from a ZIP60(1:6) as mutants with case-values greater than unity are progressively retained in the (1:1)-seeded population.

Examination of the elite genomes across the course of the 500 generations, discussed and illustrated in [13], shows that although the ZIP60(1:1) population starts out composed entirely of 1-case genomes, after a while the number of 2-case, 4-case, and 6-case mutant genomes starts to increase, and by the end of each experiment the elite individual is almost always a 6-case genome. ZIP60(1:6)-seeded experiments also virtually always end with 6-case elite genomes.

3.3 Principal Component Analysis

The results just presented demonstrate that ZIP60(1:1) most consistently out-performs ZIP8, which strongly suggests that the larger number of additional parameters are indeed useful. However, as was noted above, it is possible for a ZIP60 genome to be functionally equivalent to a lower-dimensioned ZIP*n* genome. In the most extreme case, if all the values in each homologous set of parameters are equal for any one genome (so, e.g., $\beta_{\Delta 1}=\beta_{\Delta 2}=\beta_{\Delta 3}=\beta_{\Delta 4}=\beta_{\Delta 5}=\beta_{\Delta 6}$), or if the differences between them are all sufficiently small to be ignored as mutational noise, then that ZIP60 genome is functionally equivalent to a ZIP10 genome. And if it has zero values for its $c_{a\Delta}$ and $c_{r\Delta}$ parameters, then it is effectively a ZIP8.

So, to confirm that ZIP60 is indeed an advance on ZIP8 (or ZIP10), some analysis of the final evolved parameter-sets is necessary, to see whether they contain any low-dimensional solutions embedded in higher-dimensional spaces. To this end, principal component analysis (PCA) was used on the parameter-values from the top-decile ZIP60 genomes. PCA is explained in most textbooks on multivariate analysis, e.g. [3]. Each six-dimensional homologous set of final evolved parameter values from all 18 sets of experiments was individually subjected to PCA, and the percentage of the variance in the parameter values accounted for by each principal component (PC) was calculated. If all the values in any one homologous set were equal or approximately equal, the first PC would account for very nearly 100% of the variance. However, the first PC would also account for close to 100% of the variance if the values in the homologous set were positioned along/around *any* line in the 6-D space, e.g. one where $P_{t:i} \neq P_{t:j}$ (for some P in $\{\mu, \beta, \gamma, c_a, c_r\}$; some t in $\{min, \Delta\}$; and for i, j integers in $\{1,\ldots,6\}$ with $i \neq j$). So, to identify a ZIP8/ZIP10 embedded in a ZIP60 genome, we'd need to see the first PC for each homologous set accounting for close to 100% of the variance, *and* see the angle θ between the first PC and the line $P_{t:1}=P_{t:2}=\ldots=P_{t:6}$ being very close to zero. That is for $\theta = \pi - | \cos^{-1} ((\boldsymbol{p_1} \cdot \boldsymbol{u}) / |\boldsymbol{p_1}|) - \pi |$ with $\boldsymbol{p_1}$ being the first PC (a 6-D vector) and $\boldsymbol{u}$ being a 6-D unit vector with elements $u_{i:i=1...6}$ such that $u_1=u_2=u_3=u_4=u_5=u_6>0$.

PCA analysis was performed on the entire data-set of top-decile elite genomes; and the results are presented graphically and discussed further in [14]. Although $\boldsymbol{p_1}$ accounts for more than 50% of the variance in all homologous sets, the highest value is 90.29% for $\boldsymbol{p_1}$ of the β_{min} set, which is not high enough to cause alarm. The mean variance accounted for by $\boldsymbol{p_1}$ across all homologous sets was 68% (s.d.=10%), and the minimum value was 58%. Also, the angle θ is safely high in all cases (mean=24°; s.d.=17°; min=5°; max=50°). So, the evolved ZIP60s are not ZIP8s in disguise.

3.4 Discussion: Fewer Hybrids?

Comparing the ZIP8 and ZIP60 results presented here reveals that for ZIP60 the GA much less frequently discovers hybrid values of Q_s yielding overall market dynamics that are better than those of the corresponding fixed-market CDA Q_s=0.5 experiments. That is, despite the final ZIP60 EM evolved Q_s values varying quite widely, few of them give results that are statistically significantly better than the corresponding FM results. Data tables available in [13] show that in two thirds (12 out of 18) of the original ZIP8 experiments, the EM experiment found a "hybrid" Q_s value that improved on the corresponding FM score; yet in the ZIP60 experiments, the occurrence of superior EM results fell by 67%, i.e. from 12/18 down to 4/18. This could be an indication that the previously-published results showing evolved hybrid auction mechanisms are to some extent artifacts of the lack of sophistication in the ZIP8 traders that were used in those studies. A counterargument to this is that Byde [2] presented results from applying similar GA-search for designs for hybrid sealed-bid auctions, where the GA found hybrid solutions to be preferable to the traditional first-price and second-price sealed bid auctions and those results were *independent* of the sophistication of the traders in the market. Clearly this is another issue that should be explored in more depth in future research.

4 Conclusions

From the data summarized and analyzed in this paper, it is clear that the ZIP60 variant of ZIP is a genuine improvement on the original ZIP8, and that ZIP60 parameter-vectors that outperform ZIP8 by over 10% can be found by a search/optimization process such as the simple GA used here, provided that care is taken in the progressive expansion of the dimensionality of the search-space explored by that GA. Principal component analysis of the elite evolved parameter-sets from multiple runs under differently-changing sequences of supply and demand schedules revealed that the evolved parameter-vectors make active use of considerably more values than the eight available in ZIP8. The fact that (in comparison to previous experiments using ZIP8 traders) the experiments with ZIP60 traders reported here show a reduced incidence of the discovery of "hybrid" auction mechanisms is possibly an indication that the hybrid auctions reported on in the *E-Commerce Research and Applications* journal paper [12] actually evolved as a consequence of the lack of sophistication in the behavior of ZIP8 traders: with the comparatively finer-grained responses of ZIP60 traders, hybrid mechanisms evolve much less frequently, and so it is tempting to conjecture that if the same type of auction-design experiments were repeated with even more sophisticated trader agents, hybrid mechanisms would not occur at all. Exploring that question remains one of several topics for further research.

References

[1] Bagnall, A. & Toft, I. (2005) "Autonomous Adaptive Agents for Single-Seller Sealed Bid Auctions", *Autonomous Agents & Multi Agent Systems.* In press.

[2] Byde, A. (2003), "Applying Evolutionary Game Theory to Auction Mechanism Design". *Proc. 2003 ACM Conf. on E-Commerce*. Also available as Hewlett-Packard Laboratories Technical Report HPL-2002-321.

[3] Chatfield, C. & Collins, A. (1980), *An Introduction to Multivariate Analysis.* Chapman and Hall.

[4] Clearwater, S., ed. (1995), *Market-Based Control.* World Scientific Press.

[5] Cliff, D., Harvey, I., & Husbands, P. (1993), "Explorations in Evolutionary Robotics", *Adaptive Behavior*, **2**:73–110.

[6] Cliff, D. (1997), "Minimal-intelligence agents for bargaining behaviours in market environments". Hewlett-Packard Laboratories Technical Report HPL-97-91.

[7] Cliff, D. (1998a), "Genetic optimization of adaptive trading agents for double-auction markets" in *Proceedings of Computational Intelligence in Financial Engineering (CIFEr98), New York,* IEEE/ IAFE/ Informs (preprint proceedings), pp.252–258, Apr 1998.

[8] Cliff, D. (1998b), "Evolutionary optimization of parameter sets for adaptive software-agent traders in continuous double-auction markets". Presented at the *Artificial Societies & Computational Markets (ASCMA98)* workshop at *2nd International Conference on Autonomous Agents,* May 1998. Also available as HP Labs Technical Report HPL-2001-99.

[9] Cliff, D. & Bruten, J. (1999), "Animat Market-Trading as Collective Social Adaptive Behavior". *Adaptive Behavior* **7**(3&4):385–414.

[10] Cliff, D. (2002a), "Evolution of market mechanism through a continuous space of auction-types". Presented at *Computational Intelligence in Financial Engineering (CIFEr)*, Hawaii, May 2002.

[11] Cliff, D. (2002b), "Evolution of market mechanism through a continuous space of auction-types II: Two-sided auction mechanisms evolve in response to market shocks". In: H. Arabnia, & Y. Mun. (eds) *Proc. Int. Conf. Internet Computing IC02*, Vol. III, CSREA Press, pp.682–688.

[12] Cliff, D. (2003), Explorations in evolutionary design of online auction market mechanisms. *Electronic Commerce Research & Applications Journal,* **2**(2):162–175, 2003.

[13] Cliff, D. (2005), ZIP60: *Further Explorations in the Evolutionary Design of Online Auction Market Mechanisms.* Hewlett-Packard Laboratories Technical Report HPL-2005-85.

[14] Cliff, D. (2006), *ZIP60: an enhanced extension of the ZIP trading algorithm.* Submitted for review.

[15] Das, R., Hanson, J., Kephart, J., & Tesauro, G. (2001), "Agent-human interactions in the continuous double auction" *Proc. of the Int. Joint Conference on Artificial Intelligence (IJCAI-01).*

[16] Feltovich, N. (2003), "Nonparametric tests of differences in medians: comparison of the Wilcoxon-Mann-Whitney and robust rank-order tests", *Experimental Economics,* **6**:273–297.

[17] Friedman, D. (1991), “A Simple Testable Model of Price Formation in the Double Auction Market”, *Journal of Economic Behavior & Organization* **15**:47-70.

[18] Gerding, E., Somefun, K., & La Poutré, H. (2004), “Multi-Attribute Bilateral Bargaining in a One-to-Many Setting”, In *Proc. Workshop on Agent Mediated Electronic Commerce VI (AMEC-04).*

[19] Gjerstad, S. & Dickhaut, J. (1998), “Price Formation in Double Auctions”, *Games & Economic Behavior*, **22**:1–29.

[20] Gode, D. & Sunder, S. (1993), “Allocative efficiency of markets with zero-intelligence traders”, *Journal of Politicical Economy,* **101**:119–137.

[21] Greenwald, A., Guillemette, B., Naroditskiy, V., & Tschantz, M., (2005), “Scaling Up the Sample Average Approximation Method for Stochastic Optimization with Applications to Trading Agents” in Jansen, S. (ed) *Notes IJCAI-05 Workshop on Trading Agent Design & Analysis (TADA05).*pp.21–27.

[22] Harvey, I. (1994), *The Artificial Evolution of Adaptive Behavior*. PhD Thesis, School of Cognitive and Computing Sciences, University of Sussex, U.K.

[23] He, M., Leung, H., & Jennings, N. (2003), "A fuzzy logic based bidding strategy for autonomous agents in continuous double auctions*" IEEE Transactions on Knowledge and Data Engineering* **15**(6):1345–1363.

[24] Ladley, D. & Bullock, S. (2005), “Who to listen to: Exploiting information quality in a ZIP-agent market” in Jansen, S. (ed) *Notes IJCAI-05 Workshop on Trading Agent Design & Analysis (TADA05).* pp.28-34.

[25] Li, L. & Smith, S. (2004), "Speculation Agents for Dynamic, Multi-period Continuous Double Auctions in B2B Exchanges", *Proc. 37th Hawaii International Conference on System Science*, Hawaii, January 2004.

[26] Lochner, K. & Wellman, M. (2004), “Rule-Based Specification of Auction Mechanisms.” *Proc. Third International Joint Conference on Autonomous Agents and Multiagent Systems.* pp.818-825.

[27] Pardoe, D. & Stone, P. (2005), “Developing Adaptive Auction Mechanisms”, *ACM SIGecom Exch.*, **5**(3):1-10.

[28] Park, S., Durfee, E., & Birmingham, W. (2004), “Use of Markov Chains to Design an Agent Bidding Strategy for Continuous Double Auctions”, *Journal of Artificial Intelligence Research (JAIR),* **22**: 175–214.

[29] Phelps, S., Parsons, S., & McBurney, P. (2004), “An Evolutionary Game-theoretic Comparision of two Double Auction Market Designs”. *Proceedings 4th International Workshop on Agent-Mediated E-Commerce* (AMEC-IV).

[30] Preist, C. & van Tol, M. (1998), “Adaptive Agents in a Persistent-Shout Double Auction”. *Proceedings of ICE-98*, ACM.

[31] Reeves, D., Wellman, M., Mackie-Mason, J., & Osepayshvili, A. (2005), “Exploring Bidding Strategies for Market-Based Scheduling”, *Decision Support Systems*, **39**:67–85.

[32] Robinson, N. (2002), *Evolutionary Optimization of Market-Based Control Systems for Resource Allocation in Compute Farms.* MSc Thesis, School of Cognitive & Computing Sciences, University of Sussex, U.K.

[33] Rust, J., Miller, J., & Palmer, R. (1992), “Behavior of trading automata in a computerized double auction market” in Friedman, D., & Rust, J. (eds) *The Double Auction Market: Institutions, Theories, and Evidence*, pp.155-198. Addison-Wesley;

[34] Shipp, D. (2004), *The effects of changes to supply and demand on trader agents and marketplaces.* MSc Thesis, School of Computing, University of Leeds, UK.

[35] Siegel, S., & Castellan, N. (1988), *Nonparametric Statistics for the Behavioral Sciences*, McGraw Hill.

[36] Smith, V. (1962), “Experimental study of competitive market behavior” *Journal of Political Economy,* **70**:111–137.

[37] Stanley, K. & Miikkulainen, R. (2004), “Competitive Coevolution through Evolutionary Complexification”, *Journal of Artificial Intelligence Research (JAIR),* **21**:63–100.

[38] Tesauro, G. & Das, R. (2001), “High-Performance Bidding Agents for the Continuous Double Auction”, *Economic Agents, Models, & Mechanisms Workshop, International Joint Conference on Artificial Intelligence (IJCAI-01).*

[39] Tesauro, G. & Bredin, J. (2002), “Strategic Sequential Bidding in Auctions using Dynamic Programming**”,** *Proc First Int. Conf. Autonomous Agents & Multiagent Systems (AAMAS-02).*

[40] Vytelingum, P., Dash, R., David, E., & Jennings, N. (2004), "A risk-based bidding strategy for continuous double auctions" *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI04)*, Valencia, Spain, pp.79-83.

[41] Walia, V. (2002), *Evolving Market Design*, MSc Thesis, Department of Computer Science, University of Birmingham, UK.

[42] Wellman, M., Reeves, D., Lochner, K., & Suri, R. (2005), “Searching for Walverine 2005” in Jansen, S. (ed.) *Notes IJCAI-05 Workshop on Trading Agent Design & Analysis (TADA05)* pp.1–6.

[43] Wichett, D. (2004), *Coadaptive Dynamics of Minimal Intelligence Trading Agents.* MSc Thesis, Department of Computer Science., University of Birmingham, UK.

[44] Wilson, R. (1987), "On Equilibria of Bid-Ask Markets" in Feiwel, G. (ed.) *Arrow & the Ascent of Modern Economic Theory*, pp.375-414. New York University Press.

Savings in Combinatorial Auctions Through Transformation Relationships

Andrea Giovannucci[1], Jesús Cerquides[2], and Juan A. Rodríguez-Aguilar[1]

[1] Artificial Intelligence Research Institute, IIIA-CSIC, Spain
{andrea,jar}@iiia.csic.es
[2] Dept. de Matemàtica Aplicada i Anàlisi, Universitat de Barcelona, Barcelona, Spain
cerquide@maia.ub.es

Abstract. In a previous work we extended the notion of multi-unit combinatorial reverse auction (MUCRA) by adding a new dimension to the goods at auction. A buyer can express transformability relationships among goods: some goods can be transformed into others at a transformation cost. Through this new auction type, a buyer can find out what goods to buy, to whom, and what transformations to apply to the acquired goods in order to obtain the best savings. The main focus of the paper is to perform some preliminary experiments to quantitatively assess the potential savings that a buying agent may obtain in considering transformation relationships.

1 Introduction

Since many reverse (or direct) auctions involve the buying (or selling) of a variety of complementary assets, combinatorial auctions [2], (CA) have recently deserved much attention in the literature. In particular, a significant amount of work has been devoted to the problem of selecting the set of winning bids chapters 12, 13, 14, 15, and 16 of [2]. Nonetheless, to the best of our knowledge, while the literature has considered the possibility to express relationships among goods on the bidder side —such as complementarity and substitutability—, the impact of the production relationships among the different assets to sell/buy on the bid-taker side has been only addressed so far in [3].

Consider that a company devoted to the assembly and repairing of personal computers (PCs) requires to assembly new PCs in order to fulfil his demand. Figure 1 graphically represents the way a PC is assembled. Our graphical description largely borrows from the representation of Place/Transition Nets (PTN) [6], a particular type of Petri Net. Each circle (corresponding to a PTN *place*) represents a good. Horizontal bars connecting goods represent assembly/disassembly operations, likewise *transitions* in a PTN. Assembly and disassembly operations are labelled with an indexed *t*, and shall be referred to as *transformation relationships* (t-relationships henceforth). In particular, t_1 and t_2 stand for assembly operations. An arc connecting a good to a transformation indicates that the good is an *input* to the transformation, whereas an arc connecting a transformation to a good indicates that the good is an *output* from the transformation. In our example, a CPU, RAM, USB and Empty Board are *input goods* to t_2, whereas Motherboard is an *output good* of t_2. Thus, t_2 represents the way a motherboard is manufactured (assembled). The labels on the arcs connecting *input goods* to transitions, and

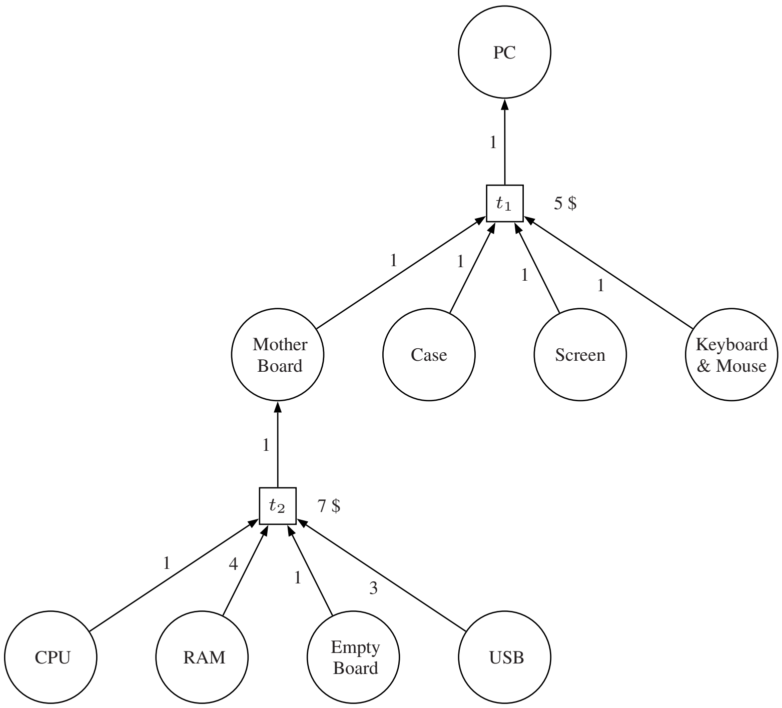


Fig. 1. Graphical representation of an RFQ with t-relationships

the labels on the arcs connecting *output goods* to transitions indicate the units required of each *input good* to perform a transformation and the units generated per *output good* respectively. In figure 1, the labels on the arcs connected to t_2 indicate that 1 motherboard is assembled from 1 CPU, 4 RAM units, 3 USBs and 1 empty motherboard. Each transformation has an associated cost every time it is carried out. In our example, assembling a motherboard via t_2 costs €7.

Say that the company's warehouse contains most of the components composing each PC. However, there are no components to assemble motherboards. Therefore, the company would have to start an automated sourcing process to acquire such components. For this purpose, it may opt for running a combinatorial reverse auction with qualified providers. But before that, a buying agent may realise that he faces a decision problem: shall he buy the required components to assemble them in house into motherboards, or buy already-assembled motherboards, or opt for a *mixed purchase* and buy some components to assemble them and some already-assembled motherboards? This concern is reasonable since the cost of components plus transformation (assembly) costs may eventually be higher than the cost of already-assembled motherboards. Hence, the buying agent requires a combinatorial reverse auction mechanism that provides: (a) a language to express required goods along with the relationships that hold among them; and (b) a winner determination solver that not only assesses what goods to buy and to whom, but also the transformations to apply to such goods in order to obtain the initially required ones. In the first part of the paper we summarize how we solved these issues in [3].

Firstly, since state-of-the-art multi-unit CA only allow buying agents to require a fixed number of units per good when expressing his requirements (henceforth referred to as *Request for Quotation* (RFQ)), we undo such constraint to allow for the introduction of t-relationships among goods. Thus, we introduce a formal definition of a *Transformability Network Structure* (TNS) that largely borrows from Place/Transition Nets [6], where transitions stand for t-relationships and places stand for goods. Secondly, we extend the formalisation of multi-unit combinatorial reverse auction (MUCRA), departing from the model in [9], to introduce transformability by applying the expressiveness power of multi-set theory. Additionally, we provide a mapping of our formal model to integer programming that takes into account t-relationships to assess the winning set of bids along with the transformations to apply in order to obtain a buying agent's initial requirements. At this point, it is important to make clear that we focus on the following central problem: given a collection of bids on bundles and a collection of t-relationships, find a set of non-conflicting bids that minimises cost. Thus, it is beyond the scope of this paper to design any CA mechanisms that consider t-relationships from a game-theoretic perspective. Thirdly, we empirically analyse the benefits of introducing t-relationships with respect to a classical multi-unit combinatorial reverse auction. We explain how to generate artificial negotiation scenarios to compare in a fair way the two types of auctions. Then, we experimentally observe that the benefits of introducing t-relationships for a buyer increase when its production process is more efficient than the providers' one, that is, when its transformation costs are smaller.

The paper is organised as follows. In section 2 we provide some background knowledge on place/transition nets and multi-sets. In section 3 we present a formal model of multi-unit combinatorial reverse auctions with t-relationships among goods, along with its winner determination problem and its mapping to integer programming. Section 4 thoroughly describes the data set generator and some experimental results. Finally, section 5 draws some conclusions and outlines directions for future research.

2 Background

A *multi-set* is an extension to the notion of set, considering the possibility of *multiple appearances* of the same element. A *multi-set* $\mathcal{M}_X$ over a set X is a function $\mathcal{M}_X : X \to \mathbb{N}$ mapping X to the cardinal numbers. For any $x \in X$, $\mathcal{M}_X(x) \in \mathbb{N}$ is called the *multiplicity* of x. An element $x \in X$ *belongs* to the multi-set $\mathcal{M}_X$ if $\mathcal{M}_X(x) \neq 0$ and we write $x \in \mathcal{M}_X$. We denote the set of multi-sets over X by X_{MS}. Given the multi-sets $\mathcal{M}_S, \mathcal{M}'_S \in S_{MS}$, their union is defined as: $\mathcal{M}_S \cup \mathcal{M}'_S(x) = \mathcal{M}_S(x) + \mathcal{M}'_S(x)$.

Following [6], a *Place/Transition Net Structure* (PTNS) is a tuple $N = (G, T, A, E)$ such that: (1) G is a set of *places*; (2) T is a finite set of *transitions* such that $P \cap T = \emptyset$; (3) $A \subseteq (G \times T) \cup (T \times G)$ is a set of *arcs*; (4) $E : A \to \mathbb{N}^+$ is an *arc expression* function. A *marking* of a PTNS is a multi-set over G. A PTNS with a given initial marking $\mathcal{M}_0 \in G_{MS}$ is denoted by $PTN = (N, \mathcal{M}_0)$ and it is called a *Place/Transition Net* (PTN). The graphical representation of a PTNS is composed of the following graphical elements: places are represented as circles, transitions are represented as bars, arcs connect places to transitions or transitions to places, and E labels arcs with values (see figure 1).

A *step* is a non-empty and finite multi-set over T. A step $\mathcal{S} \in T_{MS}$ is *enabled* in a marking $\mathcal{M} \in G_{MS}$ if the following property is satisfied: $\forall g \in G \sum_{t\in\mathcal{S}} E(g,t)\mathcal{S}(t) \leq \mathcal{M}(g)$.

Let step $\mathcal{S}$ be enabled in a marking $\mathcal{M}_1$. Then, $\mathcal{S}$ may *occur*, changing the $\mathcal{M}_1$ marking to another $\mathcal{M}_2 \in G_{MS}$ marking. Setting $Z(g,t) = E(t,g) - E(g,t)$ $\mathcal{M}_2$ is expressed as: $\forall g \in G$ $\mathcal{M}_2(g) = \mathcal{M}_1(g) + \sum_{t\in\mathcal{S}} Z(g,t)\mathcal{S}(t)$. Moreover, we say that marking $\mathcal{M}_2$ is *directly reachable* from marking $\mathcal{M}_1$ by the occurrence of step $\mathcal{S}$, and we denote it by $\mathcal{M}_1[\mathcal{S} > \mathcal{M}_2$.

A *finite occurrence sequence* is a finite sequence of steps and markings:$\mathcal{M}_1[\mathcal{S}_1 > \mathcal{M}_2 \ldots \mathcal{M}_n[\mathcal{S}_n > \mathcal{M}_{n+1}$ such that $n \in \mathbb{N}$ and $\mathcal{M}_i[\mathcal{S}_i > \mathcal{M}_{i+1}$ $\forall i \in \{1,\ldots,n\}$. $\mathcal{M}_1$ is called the *start marking*, while $\mathcal{M}_{n+1}$ is called the *end marking*. The *firing count multi-set* $\mathcal{K} \in T_{MS}$ associated to a finite occurrence sequence is the union of all its steps: $\mathcal{K} = \bigcup_{i\in\{1,2,\ldots,n\}} \mathcal{S}_i$.

A marking $\mathcal{M}''$ is *reachable* from a marking $\mathcal{M}'$ iff there exists a finite occurrence sequence having $\mathcal{M}'$ as start marking and $\mathcal{M}''$ as end marking. We denote it as $\mathcal{M}'[\mathcal{S}_1 \ \ldots \mathcal{S}_n > \mathcal{M}''$, where $n \in \mathbb{N}$. Furthermore the start and end markings are related by the following equation:

$$\forall g \in G \qquad \mathcal{M}''(g) = \mathcal{M}'(g) + \sum_{t\in\mathcal{K}} Z(g,t)\mathcal{K}(t). \tag{1}$$

The set of all possible markings reachable from a marking $\mathcal{M}'$ is called its *reachability set*, and is denoted as $R(N,\mathcal{M}')$.

In [8], Murata shows that in an *acyclic* Petri Net a marking $\mathcal{M}''$ is *reachable* from a marking $\mathcal{M}'$ iff there exists a multi-set $\mathcal{K} \in T_{MS}$ such that expression 1 holds (which is equivalent to say that the state equation associated to a PTN admits an integer solution). As a consequence, when a Petri Net is acyclic, the reachability set $R(N,\mathcal{M}')$ is represented as:

$$\begin{aligned} R(N,\mathcal{M}') = \{\mathcal{M}'' \mid \exists \mathcal{K} \in T_{MS} : \forall g \in G \\ \mathcal{M}''(g) = \mathcal{M}'(g) + \sum_{t\in\mathcal{K}} Z(g,t)\mathcal{K}(t)\}. \end{aligned} \tag{2}$$

3 MUCRA with t-Relationships

In this section we formalise the winner determination problem (WDP) for MUCRA with t-relationships (MUCRAtR) borrowing from our work in [3].

3.1 Transformability Network Structures

A Transformability Network Structure describes the different ways in which goods can be transformed and at which cost. More formally, a *transformability network structure* (TNS) is a pair $S = (N, C_T)$, where $N = (G,T,A,E)$ is a Place-Transition Net Structure and $C_T : T \rightarrow \mathbb{R}^+$ is a cost function. The cost function associates a *transformation cost* to each *t-relationship*. In this context we associate: (1) the *places* in G to a set of goods to negotiate upon; (2) the *transitions* in T to a set of *t-relationships* among goods;

(3) the *directed arcs* in A along with their weights E to the specification of the number of units of each good that are either consumed or produced by a transformation.

The values of C and the values of E label respectively transitions (between parenthesis) and arcs in figure 1.

Given a Place/Transition net $PTN = (N, \mathcal{M}_0)$, if we consider $\mathcal{M}_0$ as a good configuration, PTN defines the space of good configurations *reachable* by applying transformations to $\mathcal{M}_0$. The application of transformations is obtained by firing transitions on PTN. Hereafter, we consider the concepts of *transformation step*, *enabling of a transformation step*, *occurrence of a transformation step* and *transformation sequence* as the counterparts to, respectively, *step*, *enabling of a step*, *occurrence of a step*, and *finite occurrence sequence* on a PTN.

We also need to define the concept of transformation cost, taking into account the cost of transforming good configuration $\mathcal{M}_0$ into another good configuration $\mathcal{M}_1 \in R(N, \mathcal{M}_0)$ by means of some transformation sequence $J = (\mathcal{S}_1, \ldots, \mathcal{S}_n)$. The $\mathcal{K}$ firing count multi-set associated to J accounts for the number of times a transition in the sequence is fired. Thus, the cost of transforming good configuration $\mathcal{M}_0$ into good configuration $\mathcal{M}_1$ amounts to adding the transformation cost of each transition in the firing count multi-set $\mathcal{K}$ associated to J. We assess the transformation cost associated to J as $C_{TS}(J) = \sum_{\mathcal{S} \in J} \sum_{t \in \mathcal{S}} C_T(t)\mathcal{S}(t) = \sum_{t \in \mathcal{K}} C_T(t)\mathcal{K}(t)$. Notice that the transformation cost of a transformation sequence only depends on its firing count multi-set.

3.2 WDP for MUCRA with t-Relationships

In a classic MUCRA scenario, a *Request for Quotation* (RFQ) can be expressed as a multi-set $\mathcal{U} \in G_{MS}$ whose multiplicity indicates the number of units required per good. In the example of figure 1, if $\mathcal{U}(motherboard) = 1, \mathcal{U}(CPU) = 1, \mathcal{U}(RAM) = 2, \mathcal{U}(EmptyBoard) = 1, \mathcal{U}(USB) = 2$, $\mathcal{U}$ would be representing a buying agent's need for 1 motherboard (M), 1 CPU (C), 1 empty board (E), 2 RAM units (R), and 2 USB (U) connectors. Nonetheless, since t-relationships hold among goods, the buyer may have different alternatives depending on the bids he receives. If we represent each bid as a multi-set $\mathcal{B} \in G_{MS}$, whose multiplicity indicates the number of units offered per good, the buyer might, for instance, have the following alternatives: (1) buy all items as requested, formally $\mathcal{M}_0 = \{M, C, R, R, E, U, U\}$; and (2) $\mathcal{M}_1 = \{C, R, R, R, R, E, U, U, U\} \bigcup \{C, R, R, E, U, U\}$, do not buy any motherboard, but buy its parts (1 CPU, 4 RAM units, 1 Empty Board, and 3 USB connectors) instead to manufacture it at cost $C_T(t_2) = €7$. The overall cost of the purchase results from the cost of the acquired units plus the additional transformation cost. Notice that both alternatives allow the buyer to obtain his initial requirement, though each one at a different cost. The goal of the WDP is to assess what alternative to select optimally.

We begin by defining the set of possible auction outcomes. Given a set of bids B, a possible auction outcome is a pair (W, J), where $W \subseteq B$, and $J = (\mathcal{S}_1, \ldots, \mathcal{S}_n)$ is a transformation sequence, such that the application of J to $PTN = (N, \cup_{\mathcal{B} \in W} \mathcal{B})$ allows a buyer to obtain a good configuration that fulfils his requirements in $\mathcal{U}$. More formally, the set of possible auction outcomes is defined (assuming free disposal) as:

$$\Omega = \{(W, J), W \subseteq B \mid \exists \mathcal{X} \in G_{MS}(\bigcup_{\mathcal{B} \in W} \mathcal{B})[J > \mathcal{X}, \mathcal{X} \supseteq \mathcal{U}\}. \tag{3}$$

To each auction outcome (W, J) we associate an *auction outcome cost* as follows:

$$C_O(W, J) = \sum_{\mathcal{B} \in W} C_B(\mathcal{B}) + C_{TS}(J) \tag{4}$$

where $C_B : B \to \mathbb{R}^+$ stands for the bid cost function.

Given a set of bids B, an RFQ $\mathcal{U} \in G_{MS}$, and a transformability network structure $S = (N, C_T)$, the winner determination problem for an MUCRA with t-relationships amounts to assessing the auction outcome $(W^{opt}, J^{opt}) \in \Omega$ that minimises the auction outcome cost function C_O. Formally,

$$(W^{opt}, J^{opt}) = \underset{(W,J) \in \Omega}{\operatorname{argmin}}\, C_O(W, J) \tag{5}$$

3.3 Mapping to Integer Programming

In section 2, we defined the reachability set according to equation 2 for the case of acyclic Petri nets. Thus, if we restrict to the case of acyclic TNS, a finite occurrence sequence J is completely specified by its firing count vector $\mathcal{K}$. Then, we can rewrite expressions 3 and 4 respectively as follows:

$$\Omega = \{(W, \mathcal{K}), W \subseteq B, \mathcal{K} \in T_{MS} \mid \exists \mathcal{X} \in G_{MS}(\bigcup_{\mathcal{B} \in W} \mathcal{B})[\mathcal{K} > \mathcal{X}, \mathcal{X} \supseteq \mathcal{U}\}. \tag{6}$$

$$C_O(W, \mathcal{K}) = \sum_{\mathcal{B} \in W} C_B(\mathcal{B}) + C_{TS}(\mathcal{K}) \tag{7}$$

where $C_{TS}(\mathcal{K}) = \sum_{t \in \mathcal{K}} C_T(t)\mathcal{K}(t)$. Hence, the WDP when considering acyclic TNSs can be restated, from equation 5, to assess:

$$(W^{opt}, \mathcal{K}^{opt}) = \underset{(W,\mathcal{K}) \in \Omega}{\operatorname{argmin}}\, C_O(W, \mathcal{K}) \tag{8}$$

We can model the problem of assessing $(W^{opt}, \mathcal{K}^{opt})$ as an Integer Programming problem. For this purpose, we need to associate integer variables to the elements in: (1) a generic subset of bids ($W \subseteq B$); and (2) a generic firing count multi-set ($\mathcal{K}$).

In order to represent W we assign a binary decision variable $x_{\mathcal{B}}$ to each bid $\mathcal{B} \in B$, standing for whether $\mathcal{B}$ is selected ($x_{\mathcal{B}} = 1$) or not ($x_{\mathcal{B}} = 0$) in W. A multi-set is uniquely determined by its mapping function $\mathcal{K} : T \to \mathbb{N}$. Hence, we represent a multi-set $\mathcal{K} \in T_{MS}$ by considering an integer decision variable q_t for each $t \in T$. Each q_t represents the multiplicity of element t in the $\mathcal{K}$ multi-set.

Then, consider $G = \{g_1, \ldots, g_n\}, n \in \mathbb{N}$, is a finite set of goods, $T = \{t_1, \ldots, t_r\}$, $r \in \mathbb{N}$, is a finite set of transitions, $U = \{u_1, \ldots, u_n\}$ is a finite set of requirements, where $u_i = \mathcal{U}(g_i)$ stands for the number of units requested of good g_i, and $B = \{\mathcal{B}_1, \ldots, \mathcal{B}_m\}, m \in \mathbb{N}$, is a finite set of bids. Furthermore, for each bid $\mathcal{B}_j \in B$ we construct a pair $\langle p_j, [a_j^1, \ldots, a_j^n] \rangle$ where $p_j = p(\mathcal{B}_j)$ stands for the bid price and $a_j^i = \mathcal{M}_{\mathcal{B}_j}(g_i)$ stands for the units offered of good g_i by bid $\mathcal{B}_j$. Therefore, the problem represented by expression 8 can be translated into the following integer program:

$$min[\sum_{j=1}^{m} x_j p_j + \sum_{k=1}^{r} q_k c(t_k)] \tag{9}$$

$$s.t.\ \forall 1 \le i \le n \ \ \sum_{j=1}^{m} a_j^i x_j + \sum_{k=1}^{r} q_k m_k^i \ge u_i \tag{10}$$

where $x_j \in \{0,1\}\ \forall 1 \le j \le m$ stands for whether bid $\mathcal{B}_j$ is selected or not, and $m_k^i = Z(g_i, t_k)$. Hence, notice that each element m_k^i is in fact obtained from the *incidence matrix*[8] of the place-transition net within a TNS[1]. Notice that leaving the $q_t (t \in T)$ decision variables unbounded is utterly unrealistic because it is equivalent to say that the buyer has got the capability of applying as many transformations as required to fulfil $\mathcal{U}$.Therefore it is realistic to assume that the number of in-house transformations that he can apply are constrained. Hence, we add the following constraints: $\forall t \in T\ q_t \in \{0, 1, \ldots, max_t\}$, where $max_t \in \mathbb{N}$.

The new integer program defined by expressions 9 and 10 can be readily implemented with the aid of an optimisation library. Notice too that our integer program can be clearly regarded as an extension of the integer program we must solve for an MUCRA as formalised in [10]. Thus, the second component of expression 9 changes the overall cost as transformations are applied, whereas the second component of expression 10 makes sure that the units of the selected bids fulfil with a buyer's requirements taking into account the units consumed and produced by transformations.

4 Empirical Evaluation

The main purpose of our experiments is to empirically evaluate the benefits of introducing t-relationships in a multi-unit combinatorial reverse auction. Our experiments artificially generate different data sets, each one composed of a a TNS, a buyer's requirements, and a collection of combinatorial bids. Each data set stands for a multi-unit combinatorial reverse auction problem. We solve the WDP for each auction problem regarding and disregarding t-relationships to quantitatively assess the potential savings that a buyer/auctioneer may obtain thanks to t-relationships. In order to solve the WDP for an MUCRA, as formalised in [10], we exploit the equivalence to the multi-dimensional knapsack problem pointed out in [5]. In order to solve the WDP for an MUCRA with t-relationships we implement the integer program represented by expressions 9 and 10.

4.1 Data Set Generation

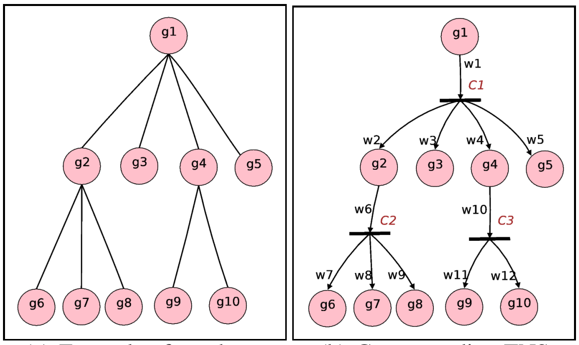
As outlined above, each data set shall be composed of: (1) a TNS; (2) an RFQ; and (3) a set of combinatorial bids. The WDP for an MUCRA will consider the last two components of the data set, whereas the WDP for an MUCRAtR will consider them all.

1 Given a TNS (N, C_T) where $N = (G, T, A, E)$ is a place-transition net with r transitions and n places, its incidence matrix $M = [m_k^i]$ is an $r \times n$ matrix of integers such that $m_k^i = E(t_k, g_i) - E(g_i, t_k)$ represents the difference of tokens of place g_i produced and consumed by transition t_k.

When we create a TNS associated to an auction, we should enforce a certain coherence among the items prices, and thus among bid prices. This concern is taken into account at bid generation time, and it will be explained in section 4.1. Thus, in the following we detail the generation of TNSs, RFQs and bids.

TNS Creation. Recall from section 3 that if we restrict to the case of an acyclic TNS, then the WDP for an MUCRAtR can be formulated as an integer program. Thus, we shall focus on generating acyclic TNSs for our data sets. For this purpose, we create TNSs fulfilling the following requirements: (a) each transition receives a single input arc; (b) each place has got no more than one input and one output arc; and (c) there exists a place, called *root place*, that has got only output arcs. Figure 2(b) depicts an example of a TNS that satisfies such requirements.

We have designed an algorithm to construct acyclic TNSs that is composed of two sequential sub-algorithms. Firstly, algorithm 1 creates a tree structure from which a second algorithm constructs a TNS by creating transformations with costs and attaching weights to the edges connecting places with transformations. Figure 2 illustrates the extension of a tree to an acyclic TNS. A distinguishing feature of our algorithm is that, since we aim at empirically assessing the potential savings when considering t-relationships independently of TNSs' shapes, it is capable of constructing acyclic TNSs that may largely differ in their shapes (e.g. with different widths, depths, either symmetric or asymmetric,...etc).



(a) Example of good tree (b) Corresponding TNS

Fig. 2. Extension of a tree to a TNS

Algorithm 1 constructs a tree of n nodes (goods) and r branching points (i.e. nodes with children). It represents the tree as a vector of n components, named $Tree$. The value of each vector component is a pointer to the index of the father good. For instance, if the i-th component of $Tree$ is set to j, it means that good g_j is the father of good g_i. Given this representation, it is easy to build a random tree. The rough idea is: (1) build a null vector $Tree$ of n components; (2) set to 0 the first component of $Tree$; (3) set each element of the vector $Tree[j]$ to a random number chosen in $[1, j-1]$. This constructive process first builds the root (g_1), then assigns a child (g_2) to the root (g_1), then assigns a child (g_3) to a random node within $\{g_1, g_2\}$,...etc. More in detail. While (line 2) the exact required number (r) of father goods are not generated, the algorithm proceeds as follows. At each iteration of the loop in line (5), the algorithm assesses the father of each good g_j. Firstly, it stores the indexes of the goods that have at least one child into $Fathers$ through the EXTRACT-NONZERO-ENTRIES function applied over $Tree$. If the father goods that have been generated are less than r (line 8), the father of g_j is randomly assessed (line 9), otherwise its father is selected out of the goods that have already children (line 11). Finally, the algorithm returns a tree (line 18).

Algorithm 1. CREATE-TREE(n, r)

```
1: $k \leftarrow 0$
2: while $k \neq r$ do
3:   $Tree \leftarrow Empty\text{-}Vector(n)$
4:   $k \leftarrow 0$
5:   for $j \leftarrow 2$ to $n$ do
6:     $Fathers \leftarrow$ EXTRACT-NONZERO-ENTRIES $(Tree)$
7:     $k \leftarrow length[Fathers]$
8:     if $k < r$ then
9:       $father \leftarrow$ EXTRACT-RANDOM-NUMBER $(1, j-1)$
10:    else
11:      $father \leftarrow$ EXTRACT-RANDOM-ELEMENT $(Fathers)$
12:    end if
13:    $Tree[j] \leftarrow father$
14:  end for
15:  $Fathers \leftarrow$ EXTRACT-NONZERO-ENTRIES $(Tree)$
16:  $k \leftarrow length[Fathers]$
17: end while
18: return $Tree$
```

From the tree generated by algorithm 1, a second algorithm extends it to generate a TNS taking as inputs the minimum and maximum arc weights (w_{min} and w_{max} respectively) and the minimum and maximum transformation costs (c_{min} and c_{max} respectively). The algorithm assigns to each branching node of the input tree a *t-relationship* having as input good the branching node, and as output goods the children of the very same branching node. Then it attaches to each created *t-relationship* a transformation cost randomly chosen in $[c_{min}, c_{max}]$. Finally, the algorithm assigns to each arc in the created TNS an integer random weight in $[w_{min}, w_{max}]$. The outputs of the algorithm are thus the incidence matrix M of the associated TNS and a transformation cost vector C. Notice that M and C are enough to characterise a TNS and to build expressions 9 and 10 of the Integer Program.

RFQ Creation. Considering the notion of requirements as described in section 3.3, an RFQ is represented as a set $U = \{u_1, \ldots, u_n\}$ where $u_i = \mathcal{U}(g_i)$ stands for the number of units requested of good g_i. We generate each number of required units $u_i \in U$ from a uniform discrete distribution $U[u_{min}, u_{max}]$, where u_{min} and u_{max} are two parameters standing for the minimum and maximum number of units required per item respectively. Notice that this differs from Leyton-Browns's approach [7] since we have not included any constraint ensuring that each data set involves the same total number of required units.

Bids Creation. In order to create a data set, the most delicate task is concerned with the generation of a collection of combinatorial bids. To the best of our knowledge, and as already pointed out in chapter 18 of [2], no real-world benchmarks of combinatorial bids do exist. We find two approaches in the CA literature to generate artificial data sets to compare winner determination algorithms for MUCRA: (1) design specific generators ([7] and chapters 18 and 19 of [2]); or (2) given the equivalence of the WDP for an MUCA/MUCRA to the multi-dimensional knapsack problem [5], employ the very same data sets used for evaluating MDKP solvers ([1]). Unfortunately, we cannot benefit from any previous results in the literature since they do not take into account the novel notion of t-relationship, and thus the generated data set never reflects the relationships among goods. For instance, consider the t_2 transformation in figure 1. It is clear that it is unrealistic that a bid offers a motherboard cheaper than some of its components. This fact motivated the need for substantially changing the approach in [7] to coherently introduce t-relationships.

In order to generate a *plausible* set of combinatorial bids, we assume that all providing agents produce goods in a *similar* manner. In other words, they share similar production structures (similar TNSs). This means that given a particular good, two providing agents will roughly use the same raw materials (components), but acquired at different prices and transformed at different costs. We believe that this assumption is utterly realistic. Under this assumption, we can artificially generate bids that can be used for both MUCRA and MUCRAtR. Nonetheless, one might be tempted to intuitively think that bids that take into account t-relationships are not valid whatsoever for an MUCRA since that would lead to an unfair comparison with MUCRAtR. Not at all. Upon reception of an RFQ, providing agents do compose bids taking into account their *own* t-relationships, and thus their *own* production costs. Thereafter, in an MUCRA scenario, the winner determination algorithm shall solely focus on finding an optimal allocation for the required goods, whereas in an MUCRAtR scenario, the winner determination algorithm shall assess whether an optimal allocation that considers the buying agent's t-relationships can be obtained. Therefore, the difference is that an MUCRAtR winner determination algorithm does consider and exploit both the buying agent's t-relationships along with the implicit transformation cost within each bid, while an MUCRA winner determination algorithm does not.

In 3.3 we mentioned that from a bid $\mathcal{B}_j \in B$ we can construct a pair $\langle p_j, [a_j^1, \ldots, a_j^n]\rangle$ where p_j stands for the bid price and $[a_j^1, \ldots, a_j^n]$ for the units offered per good. First of all, we describe how to generate the units to offer for each bid based on algorithm 2. This algorithm receives several input parameters, namely: n (number of goods); m (number of bids to generate); $p_{bid\_density}$ (parametrizes a geometric distribution [4] used for obtaining the number of goods that jointly appear in a bid); and $p_{offered\_units}$ (parameter of a geometric distribution used for obtaining the number of units offered per good.). Algorithm 2 proceeds as follows. For each bid, it firstly obtains the number of goods to jointly bid for from a geometric distribution (line 3). It subsequently obtains the number of units to offer per good (line 8) from another geometric distribution. We employed geometric distributions since they provide large variances.

Once generated the units to offer per good for all bids, we must assess all bid prices. This process is rather delicate when considering t-relationships if we want to guarantee the production of a set of plausible bids. As outlined above, we make the assumption that all providing agents in the market share similar production structures, which in turn are similar to the buying agent's one. In practice, our providing agents use the same TNS as the buying agent, though each one has his own transformation costs, which in turn are

Algorithm 2. GENERATE-OFFER-VECTOR$[n, m, p_{bid_density}, p_{offered_units}]$

```
1: for $j \leftarrow 1$ to $m$ do
2:   $[a_j^1, \ldots, a_j^n] \leftarrow Empty\text{-}Vector[n]$
3:   $k \leftarrow$ SAMPLE-GEOMETRIC-DISTRIBUTION$[p_{bid\_density}]$
4:   for $l \leftarrow 1$ to $k$ do
5:     $i \leftarrow$ randomly choose in $[1, .., n]$ such that $a_j^i = 0$
6:     $u \leftarrow 0$
7:     while $u \neq 0$ do
8:       $u \leftarrow$ SAMPLE-GEOMETRIC-DISTRIBUTION$[p_{offered\_units}]$
9:       $a_j^i \leftarrow u$
10:    end while
11:  end for
12: end for
```

assessed as a variation of the buying agent's ones. With this assumption in mind, next we describe how to produce a unitary price for each good offered in a given bid. For this purpose, we depart from the value of a parameter, p_{root}, standing for the average price of the root good of a TNS 4.1 (e.g. the root good in figure 2(b) is g_1).

The first step of our pricing policy calculates the unitary price of the root good for each bid under the assumption that all providing agents have similar values for such good. Thus, for each bid $\mathcal{B}_j \in B$, its unitary price for the root good is assessed as $P_{root,j} = p_{root} \cdot |\lambda|$, where λ is sampled from a distribution $N(1, \sigma_{root\_price})$.

After that, our pricing policy proceeds as follows. Given a particular bid and a good whose unitary price is known, this is propagated down the TNS through the transition it is linked to towards its output goods. In fact, the value to propagate results from weighting the unitary price (considering the arc connecting the input good to the transition) and adding the transformation cost of the transition. The resulting value is unevenly distributed among the output goods according to a *share factor* randomly assigned to each output good. For instance, consider the TNS in figure 2(b) and a bid $\mathcal{B}_j$ such that: its unitary cost for g_1 is $P_{g_1,j} =$ € 50, *its* transformation cost (different from the buying agent's one) for t_1 is € 10, and $w_1 = 2$. In such a case, the value to split down through t_1 towards g_2, g_3, g_4, and g_5 would be $50 * 2 + 10 =$ € 110. Say that g_2 is assigned 0.2 as share factor. Thus, $110 * 0.2 =$ € 22 would be allocated to g_2. Finally, that amount should be split further to obtain g_2 unitary price if $w_2 > 1$. For instance, if $w_2 = 2$, then the final unitary price for g_2 would be € 11.

Generalising the example above, in what follows we provide a general way of calculating the unitary price for any good in a given bid. Let $\mathcal{B}_j$ be a bid represented by $\langle p_j, [a_j^1, \ldots, a_j^n] \rangle$ and g a good such that $a_j^g \neq 0$. Let t be a transition such that g is one of its output goods, and $father(g)$ is its single input good[2]. Besides, we note as G' the set of output goods of t. Then, we obtain $P_{g,j}$, the unitary price for good g of bid $\mathcal{B}_j$ as follows:

$$P_{g,j} = \frac{P_{father(g),k} \cdot |M[father(g), t]| + c(t) \cdot |\nu|}{M[g, t]} \omega_g \tag{11}$$

where $P_{father(g),k}$ is the unitary price for good $father(g)$ in a bid $\mathcal{B}_k \neq \mathcal{B}_j$; the matrix $|M[father(g), t]|$ indicates the units of good $father(g)$ that are input to transition t; ν

[2] Recall that our method to construct acyclic TNSs ensures that there is a single input good per transition.

is a value obtained from a distribution $N(\mu_{production\_cost}, \sigma_{production\_cost})$ that weighs transformation cost $c(t)$; $M[g,t]$ indicates the number of units of good g that are output by transition t; and ω_g is the share factor for good g.

Several remarks apply to equation 11. Firstly, the share factors for output goods must satisfy $\sum_{g' \in G'} \omega_{g'} = 1$. Secondly, it may surprise the reader to realise that the value to propagate down the TNS ($P_{father(g),k}$) is collected from a different bid. We enforce this crossover operation among bids to avoid undesirable *cascading effects* that occur when we start out calculating unitary prices departing from either high or low unitary root prices. In this way we avoid to produce non-competitive and extremely competitive bids respectively that could be in some sense regarded as noise that could eventually lead to diverting results. Notice that after applying our pricing policy we obtain P, an $n \times m$ matrix storing all unitary prices.

Finally, from equation 11 we can readily obtain the bid price for a each bid $\mathcal{B}_j \in B$ as $p_j = \sum_{i=1}^{n} a_j^i \cdot P_{i,j}$. To summarise, the parameters that is possible to set when creating an MUCRAtR scenario are listed along the first column of table 1.

Table 1. Parameters characterising our experimental scenario

Parameter	***Explanation***	***Value***
n	The number of items	20
r	The number of transitions	8
u_{min}, u_{max}	The minimum/maximum number of units required per item	10/10
w_{min}, w_{max}	Minimum/Maximum arc weight	1/5
c_{min}, c_{max}	Minimum/Maximum Transformation cost	10/10
m	The number of bids to generate	1000
$p_{offered\_goods}$	Statistically sets the number of items simultaneously present in a bid	$\{0.2, 0.3, 0.4, 0.5\}$
$p_{offered\_units}$	Statistically sets the number of unit offered per item	$\{0.2, 0.3, 0.4, 0.5\}$
p_{root}	Average price of the $root$ good	1000
μ_{root_price} σ_{root_price}	Parameters of a Gaussian distribution weighting the $root$ price p_{root}	1 0.01
$\mu_{production\_cost}$ $\sigma_{production\_cost}$	Parameters of a Gaussian distribution setting the production costs difference between buyer and providers	0.8:0.1:1.8 0.1

4.2 Experimental Settings and Results

In order to measure the benefits provided by the introduction of t-relationships among goods we compute the cost of the optimal outcome, that is, the cost of the winning bid set for MUCRA (OC^{MUCRA}) and the cost of the winning bid set plus transformations for MUCRAtR ($OC^{MUCARtR}$). We define the savings increment (SI) as: $SI = 100 * \frac{OC^{MUCRA} - OC^{MUCRAtR}}{OC^{MUCRA}}$ The larger the index, the higher the benefits that a buyer can expect to obtain by using an MUCRAtR instead of an MUCRA.

The third column of table 1 shows the parameter configuration used in our experiments. In this preliminary experiment we consider the outcomes produced when varying three parameters, namely $\mu_{production\_cost}$, $p_{offered\_units}$ and $p_{bid\_density}$. In particular $\mu_{production\_cost}$ takes on values in $[0.8, 1.8]$, and we set $p_{offered\_units} = p_{bid\_density} \in \{0.2, 0.3, 0.4, 0.5\}$. Our experimental hypothesis are: (1) the SI index increases as the buyer's transformation costs decrease (larger $\mu_{production\_cost}$) with respect to the providers' ones; and (2) SI will increase when increasing the bid density (low $p_{bid\_density}$) and the number of offered units (low $p_{offered\_units}$). This hypothesis is motivated by the following reasons. Firstly, increasing the $\mu_{production\_cost}$ parameter models the fact that in-house transformations are cheaper, therefore more likely to be employed. An MUCRAtR improves savings with respect to an MUCRA as more in-house transformations are employed. In such a case the sets of winning bids of MUCRA and MUCRAtR largely differ among them. Secondly, when increasing the bid density and the number of offered units, it is very difficult for an MUCRA to allocate offers so that they perfectly fit the requirements. An MUCRAtR, instead, can employ the *free-disposal* goods obtained by *imperfect allocations* as inputs to transformations, so as to obtain other required goods. Therefore, we will analyse how the difference in production costs between the buyer and providers affects SI. The difference among the buyer's transformation costs and the average transformation costs of providers is set by the $\mu_{production\_cost}$ parameter. We expect that, as the average transformation costs of providers increases with respect to the buyer's ones, so do the benefits of using an MUCRAtR instead of an MUCRA. In fact, the experimental results do strongly agree with our hypothesis: an increment in $\mu_{production\_cost}$ causes an increment in the savings. Figure 3 depicts the results when varying $\mu_{production\_cost}$ from 0.8 to 1.8. The x axis represents the values of the $\mu_{production\_cost}$ parameter. The y axis represents the corresponding values of SI. Each point is obtained by averaging 30 runs of the experiment. The legend lists the value of the $p_{offered\_goods} = (p_{offered\_units})$ parameter[3]. As $\mu_{production\_cost}$ increases, so do savings. As the bids' density and the number of offered units jointly increase, so does savings.

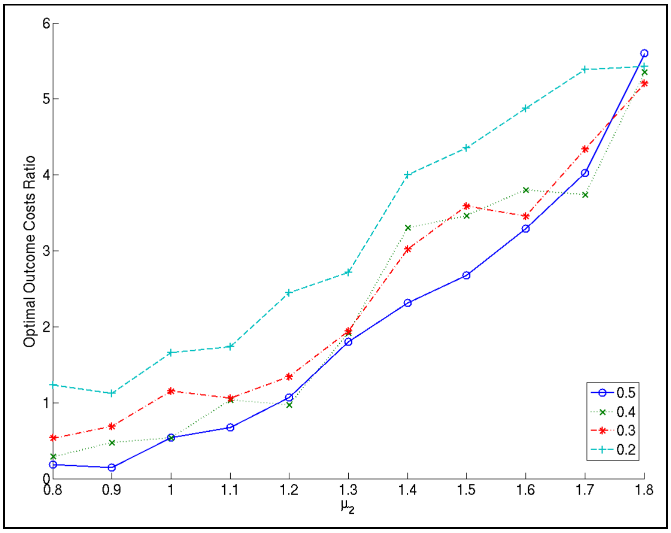


Fig. 3. Varying the $\mu_{production\_cost}$ parameter

[3] Notice that an increment in $p_{offered\_units} = p_{bid\_density}$ stands for a decrement in the number of offered units and in the bids' density, since they are parameters of a geometric distribution.

5 Conclusions and Future Work

We have performed a set of preliminary experiments to quantitatively assess the potential savings of employing an MUCRAtR instead of an MUCRA in different scenarios. The main conclusion that stems from the experiments is that the cheaper the in-house t-relationships available to the buyer, the more he can benefit from applying them. The second conclusion is that when the number of offered units and the bid density increase simultaneously, so does the savings (SI).

The novel idea presented in the paper opens several paths to future development. The first being a further development of the empirical experiments and to carry out a sound sensitivity analysis. That would allows us to fully characterize the auction scenarios where exploiting t-relationships is expected to bring larger savings than an MUCRA. On the other hand, we aim at comparing MUCRAtR and MUCRA in terms of computational complexity, performing some scalability experiments, too.

Finally, notice that it was beyond the scope of this paper any mechanism design analysis. That is left out for future work.

Acknowledgements. This work was partially funded by the Spanish Science and Technology Ministry as part of the Web-i-2 project (TIC-2003-08763-C02-00). Giovannucci Andrea enjoys the BEC.09.01.04/05-164 CSIC scholarship.

References

1. P. Chu and J. Beasley. A genetic algorithm for the multidimensional knapsack problem. *Journal of Heuristics*, 4:63–86, 1998.
2. P. Cramton, Y. Shoham, and R. Steinberg, editors. *Combinatorial Auctions*. MIT Press, 2006.
3. A. Giovannucci, J. A. Rodrguez-Aguilar, and J. Cerquides. Multi-unit combinatorial reverse auctions with transformability relationships among goods. In *LNCS*, volume 3828, Hong Kong, China, December 2005. www.iiia.csic.es/∼andrea/papers/WINE318a.pdf.
4. R. V. Hogg and J. Ledolter. *Engineering Statistics*. MacMillan Publishing Company, 1987.
5. R. C. Holte. Combinatorial auctions, knapsack problems, and hill-climbing search. In *Lecture Notes in Computer Science*, volume 2056. Springer-Verlag, Heidelberg.
6. K. Jensen. *Coloured Petri Nets, Basic Concepts, Analysis Methods and Practical Use*, volume 1, chapter 2, pages 78–80. Springer, 1997.
7. K. Leyton-Brown, Y. Shoham, and M. Tennenholtz. An algorithm for multi-unit combinatorial auctions. In *Proceedings of the AAAI Conference*, pages 56–61, 2000.
8. T. Murata. Petri nets: Properties, analysis and applications. In *IEEE*, volume 77, pages 541–580, 1989.
9. M. H. Rothkopf, A. Pekec, and R. M. Harstad. Computationally manageable combinational auctions. *Management Science*, 44(8):1131–1147, 1998.
10. T. Sandholm, S. Suri, A. Gilpin, and D. Levine. Winner determination in combinatorial auction generalizations. In *AAMAS '02*, pages 69–76, New York, NY, USA, 2002. ACM Press.

On Efficient Procedures for Multi-issue Negotiation

Shaheen S. Fatima[1], Michael Wooldridge[1], and Nicholas R. Jennings[2]

[1] Department of Computer Science
University of Liverpool, Liverpool L69 3BX, U.K
{shaheen,mjw}@csc.liv.ac.uk
[2] School of Electronics and Computer Science
University of Southampton, Southampton SO17 1BJ, U.K
nrj@ecs.soton.ac.uk

Abstract. This paper studies bilateral, multi-issue negotiation between self interested agents with deadlines. There are a number of procedures for negotiating the issues and each of these gives a different outcome. Thus, a key problem is to decide which one to use. Given this, we study the three main alternatives: the *package deal*, the *simultaneous procedure*, and the *sequential procedure*. First, we determine equilibria for the case where each agent is uncertain about its opponent's deadline. We then compare the outcomes for these procedures and determine the one that is optimal (in this case, the package deal is optimal for each party). We then compare the procedures in terms of their time complexity, the uniqueness and Pareto optimality of their solutions, and their time of agreement.

1 Introduction

Negotiation is a process that allows disputing agents to decide how to divide the gains from cooperation [13,10]. Now, in practice, most negotiations involve multiple issues. However, for such encounters, the outcome depends on the *procedure* that is used [6]. Such procedures specify how the issues will be settled. Broadly speaking, there are three possibilities: (i) Discuss the issues together as a *package deal* (PD). This gives rise to the possibility of making tradeoffs across issues. (ii) Discuss the issues simultaneously, and independently of each other. This is called the *simultaneous procedure* (SIM). (iii) Discuss the issues one after another. This is called the *sequential procedure* (SEQ). Note that in the latter two cases, the issues are settled independently and so the agents cannot make tradeoffs.

As the three procedures yield different outcomes [7], a key problem for the agents is to decide what procedure they should use. Moreover, in many practical cases the agents have to decide this in the presence of time constraints and uncertain information. Given this, it is important to study the strategic behaviour of agents in such circumstances, and to determine what is the *optimal* procedure (i.e., the one that maximises expected utilities). To this end, this paper studies and compares the three main procedures for agents with deadlines and where each agent is uncertain about the other's deadline. We show that, for each agent, the PD is the best. We then compare the procedures in terms of four important attributes: their time complexity, whether their solutions are Pareto optimal, the uniqueness of their solutions, and their time of agreement. Our analysis shows that, only the PD generates a Pareto optimal outcome, and that all three procedures have polynomial time complexity. In terms of the time of agreement, the PD and the SIM procedures are similar but the SEQ procedure is comparatively slower. Finally, we find the conditions for uniqueness of the solution.

There has been some formal comparison of different procedures to find the optimal one (see Section 5). However, all this work has two major limitations. First, it has focused on comparing procedures for negotiation without deadlines. But we believe deadlines are an important feature of most automated negotiations. Moreover, the strategic behaviour of agents with deadlines differs from that without. Second, it has only focused on finding the optimal procedure, but has not compared the solution properties of different procedures. Again, we believe this is a serious shortcoming that we rectify in this paper. Given this, our paper therefore makes a twofold contribution. First, we obtain the equilibrium for each procedure[1] when there are deadlines. Second, on the basis of this equilibrium, we provide the first comprehensive comparison of their solution properties (viz. time complexity, Pareto optimality, uniqueness, and time of agreement) and thereby allow agents to make a more informed choice about which procedure is most suitable in which circumstances.

The remainder of the paper is organised as follows. Section 2 introduces single-issue negotiation. Section 3 studies the three multi-issue procedures for the complete information scenario. Section 4 treats the agents' deadlines as uncertain. Section 5 discusses related work and Section 6 concludes.

2 Single-Issue Negotiation

We first give a reasonable standard model of single-issue negotiation and then move to the multi-issue case which is the main focus of this work. Two agents (a and b) negotiate over a single issue i using Rubinstein's alternating offers protocol [12]. Each agent has time constraints in the form of deadlines and discount factors. Since we focus on competitive scenarios with self-interested agents, we model negotiation with the 'split the pie game'. This complete information game is based on the split the pie game analysed in [14,16]. The issue i is a 'pie' of size 1 and the agents want to find how to split it between themselves. The pie shrinks with time, and this shrinkage is represented by a discount factor denoted $0 < \delta_i \leq 1$ for both agents. At time $t = 1$, the size of the pie is 1, but at $t > 1$, the pie shrinks to δ_i^{t-1}. Let $n^a \in \mathbb{N}^+$ ($n^b \in \mathbb{N}^+$) denote agent a's (b's) deadline. If an agreement is not reached by an agent's deadline, then it quits and negotiation ends in a conflict. Both agents prefer an agreement to a conflict. Hence, negotiation must end by the earlier deadline $n = min(n^a, n^b)$.

We denote the set of real numbers as $\mathbb{R}$ and the set of real numbers in the interval $[0, 1]$ as $\mathbb{R}_1$. Let $[x_i^t, y_i^t]$ denote the offer made at t where x_i^t and y_i^t denote a's and b's share respectively. Then, the set of possible offers is $\{[x_i^t, y_i^t] : x_i^t \geq 0,\ y_i^t \geq 0,\ and\ x_i^t + y_i^t = \delta_i^{t-1}\}$ where $x_i^t \in \mathbb{R}_1$ and $y_i^t \in \mathbb{R}_1$. At time $t \leq n$, if a and b receive a share of x_i^t and y_i^t (where $x_i^t + y_i^t = \delta_i^{t-1}$), then their utilities are x_i^t and y_i^t

[1] Note that existing work has obtained equilibrium for negotiation with deadlines but only for the single issue case, and a special type of the SEQ procedure for multiple issues.

respectively. Agent a (b) gets zero utility if $t > n^a$ ($t > n^b$). Finally, the conflict utility is zero for both agents.

For this setting, the offers are determined as follows. Let a make an offer at $t = 1$. To begin, let the earlier deadline is $n = 1$. If b accepts at a's offer at $t = 1$, the division occurs as agreed; if not, neither agent gets anything (since $n = 1$). Here, a is in a powerful position and is able to propose to keep 100 percent of the pie and give nothing to b.[2] Since $n = 1$, b accepts this offer and an agreement takes place at $t = 1$.

Now consider the case where the earlier deadline is $n = 2$. At $t = 1$, the size of the pie is 1 but it shrinks to δ_i at $t = 2$. In order to decide what to offer in the first round, a looks ahead to $t = 2$ and reasons backwards. It reasons that if negotiation proceeds to $t = 2$, b will take 100 percent of the shrunken pie by offering $[0, \delta_i]$ and leave nothing for a. Thus, at $t = 1$, if a offers b anything less than δ_i, b will reject the offer. Hence, at $t = 1$, a offers $[1 - \delta_i, \delta_i]$. Agent b accepts and an agreement occurs at $t = 1$.

In general, if the earlier deadline is n, a decides what to offer at $t = 1$ by looking ahead to $t = n$ and then reasoning backwards. This decision making leads a to offer $[\Sigma_{j=0}^{n-1}((-1)^j\delta_i^j), 1 - \Sigma_{j=0}^{n-1}((-1)^j\delta_i^j))]$ at $t = 1$. Agent b accepts and negotiation ends at $t = 1$. We now extend this single-issue model to the multi-issue case.

3 Multi-issue Negotiation with Complete Information

As mentioned in Section 1, the existing literature does not analyse the multi-issue procedures for negotiation with deadlines.[3] Hence, we first analyse the complete information setting. Here, a and b negotiate over $m > 1$ issues. These issues are m distinct pies and the agents want to determine how to split each one. As before, each pie is of size 1. Let the discount factor for issue c where $1 \leq c \leq m$ be $0 < \delta_c \leq 1$. For each issue, let n^a (n^b) denote agent a's (b's) deadline. In the offer for time period t, a's (b's) share for each of the m issues is represented as an m element vector $x^t \in \mathbb{R}_1^m$ ($y^t \in \mathbb{R}_1^m$). Thus, if a's share for issue c at time t is x_c^t, then b's share is $y_c^t = (\delta_c^{t-1} - x_c^t)$. The shares for a and b are together represented as the package $[x^t, y^t]$.

An agent's *cumulative utility* from the package $[x^t, y^t]$ is the sum of its utilities for each of the m issues. Let $U^a : \mathbb{R}_1^m \times \mathbb{R}_1^m \times \mathbb{N}^+ \rightarrow \mathbb{R}$ and $U^b : \mathbb{R}_1^m \times \mathbb{R}_1^m \times \mathbb{N}^+ \rightarrow \mathbb{R}$ denote the cumulative utilities for a and b respectively at time $t \leq n$ where $U^a([x^t, y^t], t) = \Sigma_{c=1}^m k_c^a x_c^t$ and $U^b([x^t, y^t], t) = \Sigma_{c=1}^m k_c^b y_c^t$ where $k^a \in \mathbb{R}^m$ denotes an m element vector for a and $k^b \in \mathbb{R}^m$ that for b. These vectors indicate how the agents value different issues. For example, if $k_c^a > k_{c+1}^a$, then agent a values issue c more than issue $c + 1$. Likewise for agent b. Each agent has complete information about all the negotiation parameters (i.e., n^a, n^b, m, k_c^a, k_c^b, and δ_c for $1 \leq c \leq m$). For this setting, we now obtain the equilibrium for the PD, the SIM, and the SEQ procedures.

[2] It is possible that b may reject such a proposal. In practice, a will have to propose an offer that is just enough to induce b to accept. However, to keep the exposition simple, we assume that a can get the whole pie by making the 100 percent proposal.

[3] The existing literature only analyses the case where each issue is discussed sequentially one after another (this is a special case of the procedures we study here). Section 5 gives details.

The package deal procedure. For this procedure, the agents use the same protocol as for the single-issue case (described in Section 2). However, an offer for the PD includes a proposal for each of the m issues. Agents are allowed to either accept a complete offer (i.e., all m issues) or reject a complete offer. An agreement can therefore take place either on all the m issues or none of them. As per single-issue negotiation, an agent decides what to offer by backward reasoning. However, since an offer for the PD includes a share for all the m issues, agents can now make tradeoffs across the issues in order to maximise their cumulative utilities. The function TRADEOFFA is agent a's function for making tradeoffs, and is described in more detail in the proof of Theorem 1. The function TRADEOFFB for b can be defined analogously.

The equilibrium offer for issue c at time t is denoted as $[a_c^t, b_c^t]$, where a_c^t and b_c^t denote the shares for a and b. We denote the equilibrium package at time t as $[a^t, b^t]$ where $a^t \in \mathbb{R}_1^m$ ($b^t \in \mathbb{R}_1^m$) is an m element vector that denotes a's (b's) share for each of the m issues. Also, $\delta^{t-1} \in \mathbb{R}^m$ is an m element vector that represents the sizes of the m pies at time t. The symbol **0** denotes an m element vector of zeroes. For each pie, the sum of the agents' shares at time t is equal to the size of the pie at t (i.e., for $1 \leq t \leq n$, $a_c^t + b_c^t = \delta_c^{t-1}$). Finally, for time period $t \leq n$, we let A(t) and B(t) denote the equilibrium strategy for agents a and b respectively. Given this, Theorem 1 characterises the equilibrium for the PD.

Theorem 1. *The following strategies form a Nash equilibrium. For $t = n$ they are:*

$$\text{A}(n) = \begin{cases} \textit{OFFER}\ \ [\delta^{n-1}, \boldsymbol{0}] & \textit{if } a\textit{'s turn} \\ \textit{ACCEPT} & \textit{if } b\textit{'s turn} \end{cases} \tag{1}$$

$$\text{B}(n) = \begin{cases} \textit{OFFER}\ \ [\boldsymbol{0}, \delta^{n-1}] & \textit{if } b\textit{'s turn} \\ \textit{ACCEPT} & \textit{if } a\textit{'s turn} \end{cases} \tag{2}$$

For $t < n$, if $[x^t, y^t]$ denotes the offer made at time t, then the strategies are:

$$\text{A}(t) = \begin{cases} \textit{OFFER}\ \ \text{TRADEOFFA}(\text{UB}(t)) & \textit{if } a\textit{'s turn} \\ \textit{If}\ \ (U^a([x^t, y^t], t) \geq \text{UA}(t))\ \textit{ACCEPT else REJECT} & \textit{if } b\textit{'s turn} \end{cases} \tag{3}$$

$$\text{B}(t) = \begin{cases} \textit{OFFER}\ \ \text{TRADEOFFB}(\text{UA}(t)) & \textit{if } b\textit{'s turn} \\ \textit{If}\ \ (U^b([x^t, y^t], t) \geq \text{UB}(t))\ \textit{ACCEPT else REJECT} & \textit{if } a\textit{'s turn} \end{cases} \tag{4}$$

where $\text{UA}(t) = U^a([a^{t+1}, b^{t+1}], t+1)$ *and* $\text{UB}(t) = U^b([a^{t+1}, b^{t+1}], t+1)$. *An agreement takes place at $t = 1$.*

Proof. *We look ahead to the last time period (i.e., $t = n$) and then reason backwards. If negotiation reaches the deadline (n), then the offering agent takes everything and its opponent gets nothing. Hence, we get Equations 1 and 2.*

In all the preceding time periods ($t < n$), the offering agent proposes a package that gives its opponent a cumulative utility equal to what the opponent would get from its own equilibrium offer for the next time period. During time period t, either a or b could be the offering agent. Consider the case where a makes an offer at t. The package that a offers at t gives b a cumulative utility of $U^b([a^{t+1}, b^{t+1}], t+1)$. However, since there is more than one issue, there is more than one package that gives b this cumulative utility.

Between these packages, a offers the one that maximises its own cumulative utility. Thus, a's tradeoff problem is to find a package $[a^t, b^t]$ that maximises $\Sigma_{c=1}^{m} k_c^a a_c^t$ such that $\Sigma_{c=1}^{m}(\delta_c^{t-1} - a_c^t)k_c^b = U^b([a^{t+1}, b^{t+1}], t+1)$ and $0 \leq a_c^t \leq 1$ for $1 \leq c \leq m$. This tradeoff problem is similar to the fractional knapsack problem [11,3], the optimal solution for which can be found using the greedy *approach (i.e., by filling the knapsack with items in their decreasing order of value per unit weight). The items in the knapsack problem are analogous to the issues in our case. The only difference is that the fractional knapsack problem starts with an empty knapsack and aims at filling it with items so as to maximise the cumulative value, while an agent's tradeoff problem can be viewed as starting with the agent having 100 per cent of all the issues and then aiming to give away portions of issues to the other agent so that the latter gets a given cumulative utility while the resulting loss in the former's utility is minimised. Hence, in order perform tradeoffs, agent a considers k_c^a/k_c^b for $1 \leq c \leq m$ because k_c^a/k_c^b is the utility that a needs to give up in order to increase b's utility by one. Since a wants to maximise its own utility and give b a utility of $U^b([a^{t+1}, b^{t+1}], t+1)$, it divides the m pies such that it gets the maximum possible share for those issues for which k_c^a/k_c^b is high and gives to b the maximum possible share for those issues for which k_c^a/k_c^b is low. Thus, a begins by giving b the maximum possible share for the issue with the lowest k_c^a/k_c^b. It then does the same for the issue with the next lowest k_c^a/k_c^b and repeats this process until b's cumulative utility is $U^b([a^{t+1}, b^{t+1}], t+1)$. In this way, agent a performs tradeoffs with the* TRADEOFFA(UB(t)) *function that uses the greedy approach described above. Thus we get Equation 3.*

Analogously, if b offers at t, we get the equilibrium package of Equation 4. In this way, the first mover obtains the offer for $t = 1$ which its opponent accepts. □

Theorem 2. *For the PD, the time to find an equilibrium offer for $t = 1$ is $\mathcal{O}(mn)$.*

Proof. *The time to compute the equilibrium offer for $t = n$ is linear in the number of issues (see Equations 1 and 2). For $t < n$, the agents make tradeoffs. Recall from Theorem 1, that an agent's tradeoff problem is analogous to the fractional knapsack problem. Hence the time complexity* TRADEOFFA *(and* TRADEOFFB*) is $\mathcal{O}(m)$ (see [11,3] for the complexity of the fractional knapsack problem). Tradeoffs are made in every time period from the $(n-1)$th to the first. Hence the time complexity of finding an offer for $t = 1$ is $\mathcal{O}(mn)$.* □

Theorem 3. *The PD has a unique equilibrium outcome if the following condition (C_1) is true:*

$$C_1: \qquad \text{for all } i \text{ and } j, \text{ if } (i \neq j) \text{ then } (k_i^a/k_i^b \neq k_j^a/k_j^b)$$

Proof. *Consider a time period $t < n$ and let a denote the offering agent. Recall from Theorem 1 that a splits the m issues in the increasing order of k_i^a/k_i^b. Thus, for a given i and j, if $k_i^a/k_i^b = k_j^a/k_j^b$, then agent a is indifferent between which of the two issues (i and j) it splits up first. For example, if $m = 2$, $n = 2$, $\delta = 0.5$, $k_1^a = 1$, $k_2^a = 2$, $k_1^b = 2$, and $k_2^b = 4$, then $k_1^a/k_1^b = k_2^a/k_2^b = 0.5$. If a is the offering agent at $t = 1$, it can offer $(1, 0)$ for issue 1 and $(1/4, 3/4)$ for issue 2. This gives a cumulative utility of 1.5 to a and 3 to b. Alternatively, a can offer $(0, 1)$ for issue 1 and $(3/4, 1/4)$ for issue 2 since this also results in the same cumulative utilities to a and b.*

But if $k_i^a/k_i^b \neq k_j^a/k_j^b$, then a splits issue i first if $k_i^a/k_i^b < k_j^a/k_j^b$ and issue j first if $k_i^a/k_i^b > k_j^a/k_j^b$. Hence there is only one possible offer that a can make at any time $t < n$. Likewise there is one possible offer that b can make at any time $t < n$. Since there is a unique offer for each time period, the equilibrium outcome is unique. □

Theorem 4. *The PD generates a Pareto optimal outcome.*

Proof. *As we consider competitive negotiations, for an individual issue c (where $1 \leq c \leq m$), an increase in one agent's utility results in a decrease in that of the other. However, for the PD procedure, an agent considers its cumulative utility from all m issues. Consequently, during the process of backward reasoning, at time $t < n$, the agent that makes tradeoffs maximises its own cumulative utility without lowering that of its opponent (with respect to what the opponent would offer in the next time period). Hence the equilibrium outcome for the PD is Pareto optimal.* □

The SIM procedure. Here the m issues are partitioned into $\mu > 1$ disjoint subsets. For $1 \leq c \leq \mu$, S_c denotes the cth partition, where $\cup_{c=1}^{\mu} S_c = \{1, \ldots, m\}$. Negotiation for each partition starts at $t = 1$ and each partition is settled using the PD. Thus, for $\mu = m$, all m issues are settled simultaneously and independently of each other. At the other extreme, for $\mu = 1$, we have only one partition which is the PD procedure described earlier. Since the issues in each subset are settled using the PD, the equilibrium for each of these μ partitions is obtained from Theorem 1. Hence we get the following results.

First, an agreement for each issue occurs at $t = 1$. Since negotiation for each partition starts at $t = 1$ and an agreement for the PD occurs at $t = 1$ (see Theorem 1), an agreement for the SIM procedure (for each partition and hence each issue) occurs at $t = 1$. Second, if $|S_c|$ is the number of issues in S_c and n is the earlier deadline then the time to determine an equilibrium offer for $t = 1$ is $\Sigma_{c=1}^{\mu} \mathcal{O}(|S_c|n)$. Let M denote the size of the largest partition. Then, $\Sigma_{c=1}^{\mu} \mathcal{O}(|S_c|n) = \mathcal{O}(Mn)$. This is because the time to find the equilibrium offer for $t = 1$ for the PD (i.e., for $\mu = 1$) is $\mathcal{O}(mn)$ (see Theorem 2), so the time to compute equilibrium offer for $t = 1$ for the cth partition is $\mathcal{O}(|S_c|n)$. Hence, for all μ partitions, the time complexity is $\Sigma_{c=1}^{\mu} \mathcal{O}(|S_c|n)$. Third, it follows from Theorem 3 that the equilibrium outcome for the SIM procedure is unique if the condition C_1 is true for each of the μ partitions (irrespective of how the m issues are split into $\mu > 1$ partitions). Finally, as Theorem 5 shows, the SIM procedure does not always generate a Pareto optimal outcome.

Theorem 5. *The SIM procedure does not always generate a Pareto optimal outcome.*

Proof. *We show this with a counter example. Let $n = 2$, $\delta = 0.5$, $m = 3$, $\mu = 2$, $S_1 = \{1, 2\}$, $S_2 = \{3\}$, $k_1^a = 1$, $k_2^a = 2$, $k_3^a = 3$, $k_1^b = 1$, $k_2^b = 0.5$, and $k_3^b = 0.25$. Let a denote the first mover. From Theorem 1, we know that in the equilibrium for partition S_1, agent a gets a share of 0.25 for issue 1 and 1 for issue 2, and b gets a share of 0.75 for issue 1 and nothing for issue 2. For partition S_2, each agent gets a share of $1/2$. Thus, a's cumulative utility from all the three issues is 3.75 and that of b is 0.875.*

Now consider the case where all the three issues are discussed using the PD. Here, $\mu = 1$ and all other parameters remain the same. In the equilibrium outcome (i.e., the package $[(\frac{1}{8}, 1, 1), (\frac{7}{8}, 0, 0)]$), a gets a cumulative utility of 5.125 and b gets 0.875. This means that the procedure with $\mu = 2$ does not generate a Pareto optimal outcome. The *reason for this is that the PD allows tradeoffs to be made across all the m issues while the simultaneous procedure only allows tradeoffs to be made across issues within each partition but not across partitions.*

The SEQ procedure. The SEQ procedure differs from the SIM one in that the partitions are now negotiated sequentially, one after another. The issues within a subset are settled using the PD. Negotiation for the first partition starts at time $t = 1$. If negotiation for the cth (for $1 \leq c \leq \mu$) partition ends at t_c, then negotiation for the $(c+1)$th partition starts at time $t_c + 1$. Each agent gets its share for all the issues in a partition as soon as the partition is settled. Since the issues in each subset are settled using the PD, the equilibrium for each of these subsets is obtained from Theorem 1 by substituting the appropriate negotiation start times for each partition. Since negotiation for each partition ends in the same time period in which it starts, the time to settle all the m issues is μ. Note that the time complexity of the SEQ procedure is the same as the SIM one. Also, like the SIM procedure, the equilibrium for SEQ is not always Pareto optimal. Finally, the SEQ procedure has a unique outcome if the condition C_1 is true fro all the partitions.

The optimal procedure. The procedure that gives a player the maximum utility is its optimal procedure. For the SEQ procedure the equilibrium outcome strongly depends on the negotiation *agenda* (i.e., the order in which the partitions are settled). There are two ways of defining the agenda [6]: *exogenously* (i.e., before the actual negotiation over the issues begins) or *endogenously* (the agents decide what issue they will settle next during the actual process of negotiation). The agenda that gives an agent the maximum utility is its optimal one [4]. Our objective here is not to determine the optimal agenda, but to consider a given agenda and compare the outcome for the SEQ procedure for the given agenda with the outcomes for the SIM and the PD procedures, in order to find the optimal one. The following theorem characterises this procedure.

Theorem 6. *Irrespective of how the m issues are split into $\mu > 1$ partitions, the PD is optimal for both parties.*

Proof. We first show that the PD is no worse than the SIM procedure. Consider the SIM procedure for $\mu > 1$. Since the difference between the procedure with $\mu = 1$ and that with $\mu > 1$ is that the former makes tradeoffs across all the m issues, while the latter does not, each agent's utility from the former is no worse than its utility from the latter.

We now show that for a given μ (where $\mu > 1$), for each agent, the outcome for the SIM procedure is better than that for the SEQ one (irrespective of the agenda for the SEQ procedure). We do this by considering each partition. Consider the partition $c = 1$. Since negotiation for the first partition starts at $t = 1$ for both SIM and SEQ procedures, the outcome for this partition is the same for $\mu = 1$ and $\mu > 1$. Hence, for the first partition, an agent gets equal utility from the two procedures. Now consider a partition $c > 1$. Let a denote the first mover for partition c (for $2 \leq c \leq \mu$) for both SIM and SEQ procedures. For the SIM procedure, negotiation for each partition starts at $t = 1$, and an agreement also occurs at $t = 1$. But, for the SEQ procedure, negotiation for the cth partition starts at $t = c$ and results in an agreement in the same time period. Since each pie shrinks with time, each agent's cumulative utility for the SIM procedure is greater than its cumulative utility for the SEQ one. Thus, for each *agent, the PD is better than the SIM procedure, and the SIM procedure is better than the SEQ one.*

We now extend the analysis to an incomplete information setting.

4 Multi-issue Negotiation with Uncertainty About Deadlines

Here, there is uncertainty about the agents' deadlines. Both agents have a probability distribution over the possible values for n^a and n^b. Let $N \in \mathbb{N}^r$ denote a vector of r integers such that for $1 \le i \le r-1$, $N_i < N_{i+1}$. This vector represents the possible values for n^a and n^b (i.e., there are r types for a and r types for b). Let $P^a : \mathbb{N}^+ \rightarrow \mathbb{R}_1$ denote the discrete probability distribution function for n^a and $P^b : \mathbb{N}^+ \rightarrow \mathbb{R}_1$ that for n^b. The vector N and the functions P^a and P^b are common knowledge to the agents. Also, each agent knows its own type but not that of its opponent. In addition, each agent knows r, δ, k^a, k^b, and m. Since there are r possible types for each agent, we define r different cumulative utility functions for each of the two agents. If a is of type i (for $1 \le i \le r$) then its cumulative utility $U_i^a : \mathbb{R}_1^m \times \mathbb{R}_1^m \times \mathbb{N}^+ \rightarrow \mathbb{R}$ from the division specified by the package $[x^t, y^t]$ at time $t \le N_i$ is $U_i^a([x^t, y^t], t) = \Sigma_{c=1}^m k_c^a x_c^t$ and zero if $t > N_i$. For b, $U_i^b = \Sigma_{c=1}^m k_c^b y_c^t$.

The PD procedure. We know from Theorem 1, that the equilibrium outcome for the complete information setting depends on the earlier deadline n. In the present setting, since there is uncertainty about n, the equilibrium outcome now differs from that in Theorem 1. We first introduce some notation and then obtain the equilibrium.

Let $\text{A}(i, t)$ denote the equilibrium strategy for an agent a of type i at time t. Analogously, for b we have $\text{B}(i, t)$. Let $[a^t, b^t]$ denote the package offered at t in equilibrium where $a^t + b^t = \delta^{t-1}$. Also, let $\text{A}(i, j, t)$ denote the equilibrium strategy for an agent a of type i for the time period t, assuming that b is of type j. Analogously, for b we have $\text{B}(i, j, t)$.

Also, let $\text{EUA}(i, t)$ ($\text{EUB}(i, t)$) denote the cumulative utility that an agent a (b) of type i expects to get from b's (a's) equilibrium offer at time t. We let $\text{EUA}(i, j, t)$ denote agent a's expected cumulative utility from its own equilibrium offer at time t if a is of type i, assuming that b is of type j ($\text{EUB}(i, j, t)$ is defined analogously). Note that the difference between $\text{EUA}(i, t)$ and $\text{EUA}(i, j, t)$ is that the former denotes a's utility for the case where b is the offering agent at t, while the latter is a's utility for the case where a is the offering agent at t. Likewise for $\text{EUB}(i, t)$ and $\text{EUB}(i, j, t)$.

Recall that in this setting, an agent only knows its own type but not that of its opponent. Since there are r possible types for each agent, there are r possible offers an agent can make at any time period (one offer corresponding to each possible type). Between these r offers, the one that gives an agent the maximum expected cumulative utility is its optimal offer. If the cth offer ($1 \le c \le r$) gives an agent the maximum expected cumulative utility, then we say that its *optimal choice* is c. For time period t, we let $\text{OPTA}(i, t)$ ($\text{OPTB}(i, t)$) denote the optimal choice for agent a (b) of type i.

Consider $t = N_r$. For this time period, for $1 \le i \le r$, we have the following (since N_r is the largest possible value for n):

$$\text{EUA}(i, N_r) = 0 \text{ and } \quad \text{EUB}(i, N_r) = 0$$

$$\textsc{eua}(i,j,N_r) = \begin{cases} 0 & \text{if } N_i < N_r \\ P^b(N_r) \times \left(\sum_{c \quad 1}^{m} k_c^a \delta_c^{t-1}\right) & \text{if } N_i = N_r \end{cases}$$

$$\textsc{eub}(i,j,N_r) = \begin{cases} 0 & \text{if } N_i < N_r \\ P^a(N_r) \times \left(\sum_{c \quad 1}^{m} k_c^b \delta_c^{t-1}\right) & \text{if } N_i = N_r \end{cases}$$

Note that $\textsc{eua}(i,j,N_r)$ and $\textsc{eub}(i,j,N_r)$ do not depend on j because in the last time period, the offering agent gets 100 per cent of all the m pies. For $t < N_r$, we have:
$\textsc{eua}(i,t) = \textsc{eua}(i,\theta,t+1)$ and $\textsc{eub}(i,t) = \textsc{eub}(i,\lambda,t+1)$
where $\theta = \textsc{opta}(i,t+1)$ and $\lambda = \textsc{optb}(i,t+1)$.

$$\textsc{eua}(i,j,t) = \begin{cases} 0 & \text{if } N_i < t \\ \sum_{e=1}^{r}\left(F^a(i,j,e,t) \times P^b(N_e)\right) & \text{if } N_i \geq t \end{cases}$$

$$\textsc{eub}(i,j,t) = \begin{cases} 0 & \text{if } N_i < t \\ \sum_{e \quad 1}^{r}\left(F^b(i,j,e,t) \times P^a(N_e)\right) & \text{if } N_i \geq t \end{cases}$$

The function F^a takes four parameters: i, j, e, and t, and returns the utility that an agent a of type i gets from offering the equilibrium package for time t, assuming that b is of type j but b is actually of type e. Obviously, b accepts a's offer if $U_e^b(\textsc{a}(i,j,t),t) \geq \textsc{eub}(e,\gamma,t+1)$ where $\gamma = \textsc{optb}(e,t+1)$. Hence, F^a is:

$$F^a(i,j,e,t) = \begin{cases} U_i^a(\textsc{a}(i,j,t)) \text{ if } U_e^b(\textsc{a}(i,j,t)) \geq \textsc{eub}(e,\gamma,t+1) \\ \textsc{eua}(i,t+1) \text{ otherwise} \end{cases}$$

where $\gamma = \textsc{optb}(e,t+1)$. The strategy $\textsc{a}(i,j,t)$ for $t = N_j$ is:

$$\textsc{a}(i,j,t) = \begin{cases} \text{OFFER } [\delta^{n-1}, \mathbf{0}] & \text{if } a\text{'s turn} \\ \text{ACCEPT} & \text{otherwise} \end{cases}$$

and for all time periods $t < N_j$ it is:

$$\textsc{a}(i,j,t) = \begin{cases} \text{OFFER } \textsc{tradeoffa}(\textsc{eub}(j,t)) & \text{if } a\text{'s turn} \\ \text{if } U_i^a([x^t,y^t],t) \geq \textsc{eua}(i,t) \text{ ACCEPT else REJECT} & \text{otherwise} \end{cases}$$

where $[x^t, y^t]$ is the package offered at t. Analogously, F^b is:

$$F^b(i,j,e,t) = \begin{cases} U_i^b(\textsc{b}(i,j,t)) \text{ if } U_e^a(\textsc{b}(i,j,t)) \geq \textsc{eua}(e,\alpha,t+1) \\ \textsc{eub}(i,t+1) \text{ otherwise} \end{cases}$$

where $\alpha = \textsc{opta}(e,t+1)$. The strategy $\textsc{b}(i,j,t)$ for $t = N_j$ is:

$$\textsc{b}(i,j,t) = \begin{cases} \text{OFFER } [\mathbf{0}, \delta^{n-1}] & \text{if } b\text{'s turn} \\ \text{ACCEPT} & \text{otherwise} \end{cases}$$

and for all preceding time periods $t < N_j$ it is:

$$\textsc{b}(i,j,t) = \begin{cases} \text{OFFER } \textsc{tradeoffb}(\textsc{eua}(j,t)) & \text{if } b\text{'s turn} \\ \text{if } U_i^b([x^t,y^t],t) \geq \textsc{eub}(i,t) \text{ ACCEPT else REJECT} & \text{otherwise} \end{cases}$$

Thus, the optimal choices for a and b are:

$$\text{OPTA}(i,t) = \text{arg max}_{j=1}^{r} \text{EUA}(i,j,t) \tag{5}$$

$$\text{OPTB}(i,t) = \text{arg max}_{j=1}^{r} \text{EUB}(i,j,t) \tag{6}$$

We compute the optimal choice for $t = 1$ by reasoning backwards from $t = N_r$. At $t = 1$, if an agent a of type i is the offering agent, then it offers the package that corresponds to b being of type OPTA(i,1). Likewise, if an agent b of type i is the offering agent, then it offers the package that corresponds to a being of type OPTB(i,1).

But since OPTA(i,1) and OPTB(i,1) are obtained under uncertainty, an agreement may or may not occur at $t = 1$. If it does not, then the agents update their beliefs as follows. Assume an agent a of type i makes an offer at $t = 1$. If this offer gets rejected, then it means that b is not of type $\text{OPTA}(i,1)$ and so a updates its beliefs about b using Bayes' rule (excluding passed deadlines and putting all the weight of the posterior distribution of a's type over all N_i such that $i \neq \text{OPTA}(i,1)$). Now, on the basis of a's offer at $t = 1$ (say $[a^1, b^1]$), b can infer the possible types for a. Thus, b also updates its beliefs using Bayes' rule (putting all the weight of the posterior distribution of a's type over $\mathcal{N}$ where $\mathcal{N} \subseteq N$ is the set of possible types for a that can offer $[a^1, b^1]$ in equilibrium). The belief update rules for the case where b offers at $t = 1$ are analogous to the case where a offers at $t = 1$. If the offer at $t = 1$ gets rejected, then negotiation goes to the next round. At $t = 2$, the offering agent (say an agent a of type i) finds $\text{OPTA}(i,2)$ with the updated beliefs. This process of updating beliefs and making offers continues until either an agreement is reached or one of the agents quits negotiation.

Theorem 7. *If $[x^t, y^t]$ denotes the offer made at time t, then for the PD procedure, for the time period $t \leq N_r$, the following strategies form a sequential equilibrium:*

$$\text{A}(i,t) = \begin{cases} \textit{QUIT} & \textit{if } t > N_i \\ \textit{OFFER } \text{TRADEOFFA}(\text{EUB}(\psi,t)) & \textit{if } a\textit{'s turn} \\ \textit{If offer gets rejected UPDATE BELIEFS} & \\ \textit{RECEIVE OFFER and UPDATE BELIEFS} & \textit{if } b\textit{'s turn} \\ \textit{If } (U_i^a([x^t,y^t],t) \geq \text{EUA}(i,t)) \textit{ ACCEPT else REJECT} & \end{cases} \tag{7}$$

$$\text{B}(i,t) = \begin{cases} \textit{QUIT} & \textit{if } t > N_i \\ \textit{OFFER } \text{TRADEOFFB}(\text{EUA}(\phi,t)) & \textit{if } b\textit{'s turn} \\ \textit{If offer gets rejected UPDATE BELIEFS} & \\ \textit{RECEIVE OFFER and UPDATE BELIEFS} & \textit{if } a\textit{'s turn} \\ \textit{If } (U_i^b([x^t,y^t],t) \geq \text{EUB}(i,t)) \textit{ ACCEPT else REJECT} & \end{cases} \tag{8}$$

for $1 \leq i \leq r$. Here, $\psi = \text{OPTA}(i,t)$ and $\phi = \text{OPTB}(i,t)$. Negotiation ends either in an agreement or a conflict. The earliest possible time of agreement is $t = 1$.

Proof. There are r possible values for the earlier deadline, and the vector N contains these possible values in ascending order. Hence, if $i < j$, then $min(N_i, N_j)$ is N_i. To begin, consider the time period $t = 1$ and assume that an agent a of type i is the offering agent. There are r possible offers that a can make at t, one offer corresponding to each *of the possible types for b (i.e., $\text{A}(i, j, 1)$ for $1 \leq j \leq r$). From these, a offers the one that gives it the maximum expected cumulative utility (i.e., the one with $j = \text{OPTA}(i, 1)$).*

However, since $\text{OPTA}(i, 1)$ is computed under uncertainty (i.e., on the basis of expected utilities), an agreement may or may not take place at $t = 1$. If it does not, then negotiation proceeds as follows. Consider a time period t such that $1 \leq t < N_r$. Let $[x^t, y^t]$ denote the offer made at time t. The agent that receives the offer (say a) updates its beliefs using Bayes' rule (excluding passed deadlines and putting all the weight of the posterior distribution of b's type over $\mathcal{N}$ where $\mathcal{N} \subseteq N$ is the set of possible types for b that can offer $[x^t, y^t]$ in equilibrium). If the proposed offer ($[x^t, y^t]$) gets rejected, then the offering agent (say agent b of type i) updates its beliefs using Bayes' rule (putting all the weight of the posterior distribution of a's type over all N_i such that $i \neq \text{OPTA}(i, 1)$). The belief update rules for the case where a offers at time t are analogous to the above rule. Hence we get Equations 7 and 8.

We now show that the beliefs specified above are consistent. During any time period $t < N_r$, suppose the strategy profile ($\text{A}(i, t)$, $\text{B}(i, t)$) assigns probability $1 - \epsilon$ to the above specified posterior beliefs and probability ϵ to the rest of the support for the opponent's type. As $\epsilon \rightarrow 0$, the fully mixed strategy pair converges to (A, B). Also, the beliefs generated by the fully mixed strategy pair converge to the beliefs described above. Given these beliefs, strategies A and B are sequentially rational.

We show the earliest possible time of agreement is $t = 1$ with an example: let $m = 2$, $\delta = 0.5$, $N_r = 2$, $r = 2$, $N = [1, 2]$, $k^a = [1, 2]$, $k^b = [2, 1]$, $P^a(1) = 0.1$, $P^a(2) = 0.9$, $P^b(1) = 0.9$, $P^b(2) = 0.1$. Let an agent a of type 1 (i.e., $n^a = 1$) be the offering agent at $t = 1$. Since $r = 2$, a can play two possible strategies at $t = 1$: one corresponding to the case where b is of type 1 and the other to the case where b is of type 2. For the former, a's equilibrium offer at $t = 1$ is $[1, 0]$ for each issue. Hence $\text{EUA}(1, 1, 1) = 2.7$. For the latter case, a's offer at $t = 1$ is $[0.325, 0.675]$ for the first issue and $[1, 0]$ for the second one. Hence $\text{EUA}(1, 2, 1) = 2.325$. Since $\text{EUA}(1, 1, 1) > \text{EUA}(1, 2, 1)$, $\text{OPTA}(1, 1) = 1$ and a plays the former strategy. Now if b is actually of type 1, then it accepts a's offer. Thus, the earliest possible time of agreement is $t = 1$. But if b is of type 2, it rejects a's offer since it can get a higher expected utility at $t = 2$. However, since a is of type 1, negotiation ends in a conflict.

If agent a's offer at $t = 1$ gets rejected it knows that agent b is not of type $\text{OPTA}(i, 1)$. Thus the number of possible types for b is now reduced to $r - 1$. This happens every time a makes an offer that gets rejected. When negotiation reaches time period $t = 2r - 1$, there is only one possible type for b. An agreement therefore takes place at the latest by $t = 2r - 1$. However, if $n < 2r - 1$ then negotiation may end in a conflict.

Theorem 8. *The time complexity of the PD procedure is $\mathcal{O}(mr^3 T(N_r - \frac{T}{2}))$ where $T = min(2r - 1, n)$.*

Proof. *Let a be the offering agent at $t = 1$ and let N_r be even (the proof for odd N_r is analogous). We begin with the last time period and then reason backwards. Since N_r is even and a starts at $t = 1$, it is b's turn to offer in the last time period. For $t = N_r$, the time taken to find $\text{EUB}(i, j, t)$ (for a given i and j) is $\mathcal{O}(m)$ (see the definition of $\text{EUB}(i, j, t)$). Hence, the time taken to find $\text{EUB}(i, j, t)$ for all possible types of b (i.e., $1 \leq j \leq r$) is $\mathcal{O}(mr)$. Note that at this stage $\text{EUB}(j, t - 1)$ is known for $1 \leq j \leq r$.*

Now consider $t = N_r - 1$. Since N_r is even, it is a's turn to offer at $t = N_r - 1$. In order to find $\text{A}(i,t)$*, we first need to find* $\psi = \text{OPTA}(i,t)$*. From the definition for* $\text{OPTA}(i,t)$ *we know that, for a given* i*, the time to find* $\text{OPTA}(i,t)$ *depends on the time to find* $\text{EUA}(i,j,t)$ *which in turn depends on the time to find* $\text{F}^a(i,j,e,t)$*. The time taken for* $\text{F}^a(i,j,e,t)$ *depends on the time taken for* $\text{A}(i,j,t)$*. For a given* i *and a given* j*, the time taken to find* $\text{A}(i,j,t)$ *is the time taken by the function* TRADEOFFA*. Since* $\text{EUB}(j,t)$ *is already known at time* t*, the time taken by* TRADEOFFA *is* $\mathcal{O}(m)$ *(see Theorem 2 for the complexity of* TRADEOFFA*). The time taken to find* $\text{F}^a(i,j,e,t)$ *is therefore* $\mathcal{O}(m)$*. Given this, the time to find* $\text{EUA}(i,j,t)$ *(for a given* i *and* j*) is* $\mathcal{O}(mr)$*. Hence, for a given* i*, the time to find* $\psi = \text{OPTA}(i,t)$ *is* $\mathcal{O}(mr^2)$*. Consequently, for a given* i*, the time to find* $\text{A}(i,t)$ *is* $\mathcal{O}(mr^2)$*. Recall that each agent knows only its own type and not that of its opponent. Hence we need to determine* $\text{A}(i,t)$ *for all possible types of* a *(i.e., for* $1 \leq i \leq r$*). This takes* $\mathcal{O}(mr^3)$ *time. Note that at this stage* $\text{EUA}(i,j,t)$ *is known for all possible values of* i *and* j*.*

Now consider the time period $t = N_r - 2$ *when it is* b*'s turn to offer. For* $t = N_r - 2$ *and a given* i*, the time to find* $\text{OPTB}(i,t)$ *is* $\mathcal{O}(mr^2)$ *and so the time to find* $\text{OPTB}(i,t)$ *for all possible types of* b *is* $\mathcal{O}(mr^3)$*. In the same way, the computation for each time period* $t < N_r$ *takes* $\mathcal{O}(mr^3)$ *time. Hence, the total time to find the equilibrium offer for* $t = 1$ *is* $\mathcal{O}((N_r - 1)mr^3)$*. However, as noted previously, an agreement may or may not occur at* $t = 1$*. If it does not, then the agents update their beliefs and find the equilibrium offer for* $t = 2$*. The time to compute the equilibrium offer for* $t = 2$ *is* $\mathcal{O}((N_r - 2)mr^3)$*. This process of updating beliefs and finding the equilibrium offer is repeated at most* $T = min(2r - 1, n)$ *times (see the last paragraph of the proof for Theorem 7). Hence the time complexity of the PD is* $\Sigma_{i=1}^{T}\mathcal{O}((N_r - i)mr^3) = \mathcal{O}(mr^3T(N_r - \frac{T}{2}))$*.*

Obviously, Theorems 3 and 4 extend to this scenario as well.

The SIM procedure. For the SIM procedure, the equilibrium for each partition is the same as that of Theorem 7. Consequently, the time complexity of computing an equilibrium offer is $\Sigma_{c=1}^{\mu}[\Sigma_{i=1}^{T}\mathcal{O}((N_r - i)|S_c|r^3)] = \mathcal{O}(Mr^3T(N_r - \frac{T}{2}))$. As before, M denotes the size of the largest partition. It is obvious that the condition for uniqueness

Table 1. Outcomes for the incomplete information setting – m is the total number of issues and M is the number of issues in the largest partition (for a definition of C_1 see Theorem 3)

	Package deal	Simultaneous	Sequential
Time of agreement (t_c)	Earliest possible time for the cth issue $t_c = 1$ for $1 \leq c \leq m$	Earliest possible time for the cth issue $t_c = 1$ for $1 \leq c \leq m$	Earliest possible time for the cth partition $t_c = c$ for $1 \leq c \leq \mu$
Time to compute equilibrium	$\mathcal{O}(mr^3T(N_r - \frac{T}{2}))$	$\mathcal{O}(Mr^3T(N_r - \frac{T}{2}))$	$\mathcal{O}(Mr^3T(N_r - \frac{T}{2}))$
Pareto optimal?	Yes	No	No
Conditions for uniqueness	if C_1 is true	if C_1 is true for every partition	if C_1 is true for every partition

is the same as that for the SIM procedure for the complete information case. Also, the outcome is not always Pareto optimal (see Theorem 5). Finally, for each partition, the earliest possible time of agreement is $t = 1$.

The SEQ procedure. For the SEQ procedure, the equilibrium outcome for the cth (for $1 \leq c \leq \mu$) partition is obtained from Theorem 7. The condition for uniqueness is the same as that for the SEQ procedure for the complete information case. The outcome is not always Pareto optimal (see Theorem 5). Also, the time complexity of the SEQ procedure is $\mathcal{O}(Mr^3T(N_r - \frac{T}{2}))$ (see Theorem 8). Finally, for the cth partition, the earliest possible time of agreement is $t_c = c$ (since the earliest possible time of agreement for the package deal is the first time period).

The optimal procedure. For each agent, the PD is optimal. The proof is analogous to the proof for Theorem 6 (except the fact that instead of actual utilities, we now use expected utilities).

5 Related Work

A number of studies have analysed different procedures for multi-issue negotiation. For instance, Fershtman [6] extended Rubinstein's model [14] for splitting a single pie to SEQ negotiation for two pies. This model assumes complete information, imposes an agenda exogenously, and studies the relation between the agenda and the outcome of the SEQ bargaining game. On the other hand, [9,8,1] study negotiations with an endogenous agenda. For instance, [9] studies PD, SIM, and SEQ negotiation by assuming complete information. Furthermore, the agents are assumed to have discount factors but no deadlines. The main result of this work is that the PD is the optimal procedure and that for each procedure there exist multiple equilibria. [8] extends this work by finding conditions under which the equilibrium is unique. [1] developed an *asymmetric information* model for two issues and studied the PD and the SEQ procedure. A slightly different approach was taken in [2] by adding a preliminary period in which agents bargain over an agenda first and then settle the issues using this agenda. However, in [1] and [2] the players have discount factors but no deadlines. In summary, the above work differs from ours in that we consider both discount factors and deadlines, whereas previous work only considers discount factors and no deadlines.[4] Negotiation with deadlines was studied in [15] but only for a single issue. Also, the existing literature does not compare the different procedures in terms of a comprehensive list of their attributes (viz. time complexity, Pareto optimality, uniqueness, and time of agreement). Our comparative study of these attributes allows a more informed choice to be made about which procedure is most suitable in which circumstances.

Perhaps the work closest to ours is [5]. This work considers a setting which is similar to Section 4, but instead of treating the negotiation deadline as uncertain, it treats an agent's information about its opponent's utility as uncertain. For this setting, the PD is shown to be the optimal procedure.

[4] [4] only determines the optimal agenda for SEQ negotiation (with a single issue in each partition) with deadlines.

6 Conclusions and Future Work

This paper analysed the three key procedures for bilateral multi-issue negotiation between self-interested agents: the PD, the SIM, and the SEQ procedures. Our results (see Table 1) show that the PD is better than the other two because it is the optimal procedure for both agents, it is the only one to generate a Pareto optimal outcome, and it achieves these with polynomial time complexity (as the other two procedures). With regard to the time of agreement, the PD and the SIM procedures are similar in that, for the complete information setting, both procedures result in an agreement at $t = 1$ for all the issues. Also, when there is uncertainty about deadlines, the earliest possible time of agreement is the same for both procedures. But the SEQ procedure is slower in terms of the time of agreement. Finally, all the three procedures have a unique outcome under certain conditions.

In future, we will extend our symmetric information analysis by studying asymmetric information settings. Also, in this work, we modelled the players' time preferences in the form of discount factors. However, it has been shown that the outcome for negotiation with discount factors can differ from that for fixed time costs [2]. Therefore, it will be interesting to extend our analysis to negotiations with fixed time costs.

Acknowledgements

We are grateful to Sarit Kraus for her detailed comments on earlier versions of this paper.

References

1. M. Bac and H. Raff. Issue-by-issue negotiations: the role of information and time preference. *Games and Economic Behavior*, 13:125–134, 1996.
2. L. A. Busch and I. J. Horstman. Bargaining frictions, bargaining procedures and implied costs in multiple-issue bargaining. *Economica*, 64:669–680, 1997.
3. T. H. Cormen, C. E. Leiserson, R. L Rivest, and C. Stein. *An introduction to algorithms*. The MIT Press, Cambridge, Massachusetts, 2003.
4. S. S. Fatima, M. Wooldridge, and N. R. Jennings. An agenda based framework for multi-issue negotiation. *Artificial Intelligence Journal*, 152(1):1–45, 2004.
5. S. S. Fatima, M. Wooldridge, and N. R. Jennings. Muilt-issue negotiation with deadlines. *Journal of Artificial Intelligence Research*, 27:381–417, 2006.
6. C. Fershtman. The importance of the agenda in bargaining. *Games and Economic Behavior*, 2:224–238, 1990.
7. C. Fershtman. A note on multi-issue two-sided bargaining: bilateral procedures. *Games and Economic Behavior*, 30:216–227, 2000.
8. Y. In and R. Serrano. Agenda restrictions in multi-issue bargaining (ii): unrestricted agendas. *Economics Letters*, 79:325–331, 2003.
9. R. Inderst. Multi-issue bargaining with endogenous agenda. *Games and Economic Behavior*, 30:64–82, 2000.
10. S. Kraus. *Strategic negotiation in multi-agent environments*. The MIT Press, Cambridge, Massachusetts, 2001.

11. S. Martello and P. Toth. *Knapsack problems: Algorithms and computer implementations*. John Wiley and Sons, 1990.
12. M. J. Osborne and A. Rubinstein. *A Course in Game Theory*. The MIT Press, 1994.
13. J. S. Rosenschein and G. Zlotkin. *Rules of Encounter*. MIT Press, 1994.
14. A. Rubinstein. Perfect equilibrium in a bargaining model. *Econometrica*, 50(1):97–109, January 1982.
15. T. Sandholm and N. Vulkan. Bargaining with deadlines. In *AAAI-99*, pages 44–51, Orlando, FL, 1999.
16. I. Stahl. *Bargaining Theory*. Stockholm School of Economics, Stockholm, 1972.

TacTex-05: An Adaptive Agent for TAC SCM

David Pardoe, Peter Stone, and Mark VanMiddlesworth

Department of Computer Sciences
The University of Texas at Austin, Austin TX 78712, USA
{dpardoe,pstone,markvanm}@cs.utexas.edu

Abstract. Supply chains are ubiquitous in the manufacturing of many complex products. Traditionally, supply chains have been created through the interactions of human representatives of the companies involved, but advances in autonomous agent technologies have sparked an interest in automating the process. The Trading Agent Competition Supply Chain Management (TAC SCM) scenario provides a unique testbed for studying supply chain management agents. This paper introduces TacTex-05, the champion agent from the 2005 competition, focusing on its ability to adapt to opponent behavior over a series of games. The impact of this adaptivity is examined through both analysis of competition results and controlled experiments.

1 Introduction

In today's industrial world, supply chains are ubiquitous in the manufacturing of many complex products. Traditionally, supply chains have been created through the intricate interactions of human representatives of the various companies involved. However, recent advances in autonomous agent technologies have sparked an interest, both in academia and in industry, in automating the process [1] [2] [3].

One barrier to supply chain management research is that it can be difficult to benchmark automated strategies in a live business environment, due to the proprietary nature of the systems and the high cost of errors. The Trading Agent Competition Supply Chain Management (TAC SCM) scenario provides a unique testbed for studying and prototyping supply chain management agents by providing a competitive environment in which independently created agents can be tested against each other over the course of many simulations in an open academic setting.

In this paper, we describe TacTex-05, the winner of the 2005 TAC SCM competition, and focus on its ability to adapt to opponent behavior over a series of games. In particular, we describe how decisions concerning purchases at the start of a game and sales at the end of a game are influenced by observations from past games. While the start- and end-game effects observed in a TAC SCM game may appear to be artifacts of the fixed game length, there are analogues in real-world supply chains: products are frequently introduced or phased out, and analyzing the behavior of competitors when such events have occurred in the past may provide clues for the future.

The remainder of this paper is organized as follows. We first summarize the TAC SCM scenario, and then describe the design of TacTex-05. Next, we describe the methods used by TacTex-05 to adapt to past opponent behavior. Finally, we examine the impact of this adaptivity, through both analysis of competition results and controlled experiments.

2 The TAC Supply Chain Management Scenario

In this section, we provide a summary of the TAC SCM scenario. Full details are available in the official specification document.[1]

In a TAC SCM game, six agents act as computer manufacturers in a simulated economy managed by a game server. The length of a game is 220 simulated days, with each day lasting 15 seconds of real time. The game can be divided into three parts: i) component procurement, ii) computer sales, and iii) production and delivery, as expanded on in the remainder of this section.

2.1 Component Procurement

The computers are made from four components: CPUs, motherboards, memory, and hard drives, each of which come in multiple varieties. From these components, 16 different computer configurations can be made. Agents must purchase these components from a set of suppliers managed by the game server.

Agents wanting to purchase components send requests for quotes (RFQs) to suppliers indicating the *type and quantity* of components desired, the *date* on which they should be delivered, and a *reserve price* stating the maximum amount the agent is willing to pay. Agents may send at most 5 RFQs per component per supplier each day. Suppliers respond to RFQs the next day by offering a price for the requested components if the request can be satisfied. Agents may then accept or reject the offers.

Suppliers have a *limited capacity* for producing components, and this capacity varies throughout the game according to a random walk. The price offered in response to an RFQ depends on the fraction of the supplier's capacity that is free before the requested due date.

2.2 Computer Sales

Customers wishing to buy computers send the agents RFQs consisting of the *type and quantity* of computer desired, the *due date*, a *reserve price* indicating the maximum amount the customer is willing to pay per computer, and a *penalty* that must be paid for each day the delivery is late. Agents respond to the RFQs by bidding in a first-price auction: the agent offering the lowest price on each RFQ wins the order. The number of RFQs sent by customers each day depends on the level of customer demand, which fluctuates throughout the game.

[1] `http://www.sics.se/tac/tac05scmspec_v157.pdf`

2.3 Production and Delivery

Each agent manages a factory where computers are assembled. Factory operation is constrained by both the components in inventory and assembly cycles. Each day an agent must send a production schedule and a delivery schedule to the server indicating its actions for the next day. The production schedule specifies how many of each computer will be assembled by the factory, while the delivery schedule indicates which customer orders will be filled from the completed computers in inventory. Agents are required to pay a small daily storage fee for all components in inventory at the factory.

3 Overview of TacTex-05

In this section we present a high-level overview of TacTex-05. Details on specific agent components are contained in the sections that follow.[2]

Figure [1] illustrates the basic components of TacTex-05 and their interaction. There are five basic tasks a TAC SCM agent must perform: i) sending RFQs to suppliers to request components, ii) deciding which offers from suppliers to accept, iii) bidding on RFQs from customers requesting computers, iv) sending the daily production schedule to the factory, and v) delivering completed computers. We assign the first two tasks to a *Supply Manager* module, and the last three to a *Demand Manager* module. The Supply Manager handles all planning related to component inventories and purchases, and requires no information about computer production except for a projection of component use over a future period, which is provided by the Demand Manager. The Demand Manager, in turn, handles all planning related to computer sales and production. The only information about components required by the Demand Manager is a projection of the current inventory and future component deliveries, along with an estimated replacement cost for each component used. This information is provided by the Supply Manager.

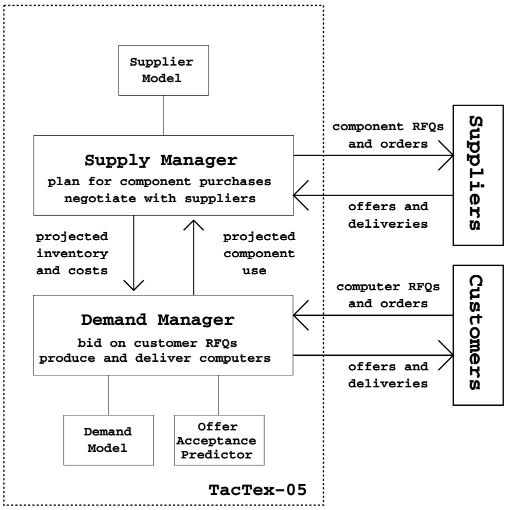


Fig. 1. Agent overview

We view the tasks to be performed by these two managers as optimization tasks: the Supply Manager tries to minimize the cost of obtaining the

[2] The information in sections 3-5 is condensed from [4] and is included to provide a fully self-contained agent description. Section 6 presents new information about the adaptive aspects of TacTex-05, the focus of this paper.

components required by the Demand Manager, while the Demand Manager seeks to maximize the profits from computer sales subject to the information provided by the Supply Manager. In order to perform these tasks, the two managers need to be able to make predictions about the results of their actions and the future of the economy. TacTex-05 uses three predictive models to assist the managers with these predictions: a predictive *Supplier Model*, a predictive *Demand Model*, and an *Offer Acceptance Predictor*.

The Supplier Model keeps track of all information available about each supplier, such as TacTex-05's outstanding orders and the prices that have been offered in response to RFQs. Using this information, the Supplier Model can assist the Supply Manager by making predictions concerning future component prices.

The Demand Model tracks customer demand and tries to estimate the underlying demand parameters With these estimates, it is possible to predict the number of RFQs that will be received on any future day. The Demand Manager can then use these predictions to plan for future production.

When deciding what bids to make in response to customer RFQs, the Demand Manager needs to be able to estimate the probability of a particular bid being accepted (which depends on the bidding behavior of the other agents). This prediction is handled by the Offer Acceptance Predictor. Based on past bidding results, the Offer Acceptance Predictor produces a function for each RFQ that maps bid prices to the predicted probability of winning the order.

4 The Demand Manager

The Demand Manager handles all computation related to computer sales and production. This section describes the Demand Manager, along with the Demand Model and the Offer Acceptance Predictor upon which it relies.

4.1 Demand Model

When planning for future computer production, the Demand Manager needs to be able to make predictions about future demand. The Demand Model is responsible for making these predictions, and does so using an approach introduced by the agent DeepMaize in 2003 (and fully described in [5]). Basically, this is a Bayesian approach that involves maintaining a probability distribution over the parameters used in the game server's algorithm for generating customer demand. Using this information, it is possible to to project expected future demand.

4.2 Offer Acceptance Predictor

In order to bid on customer RFQs, the Demand Manager needs to be able to predict the orders that will result from the offers it makes. The Offer Acceptance Predictor makes these predictions possible. For each customer RFQ received, the Offer Acceptance Predictor generates a function mapping the possible offer prices to the probability of the customer accepting the offer. (The function can thus be viewed as a cumulative distribution function.) In [6] we explored the possibility of learning to generate these functions based on past games. In TacTex-05, however, we use a simpler approach adapted from the method used by the agent Botticelli in 2003 [7]. Essentially, a linear function is generated for each computer type by performing regression on data points representing recent prices offered by TacTex-05 along with the resulting acceptance rate.

4.3 Demand Manager

The Demand Manager is responsible for bidding on customer RFQs, producing computers, and delivering them to customers. All three tasks are performed using the same greedy production scheduling algorithm. As these tasks compete for the same resources (components, completed computers, and factory cycles), the Demand Manager begins by planning to satisfy existing orders, and then uses the remaining resources in planning for RFQs. The latest possible due date for an RFQ received on the current day is 12 days in the future, meaning the production schedule for the needed computers must be sent within the next 10 days. The Demand Manager thus always plans for the next 10 days of production.

The Demand Manager begins each day by initializing its production resources using the values provided by the Supply Manager. The production scheduler is then applied to existing orders, and orders that are due immediately and can be filled from inventory are scheduled for delivery.

Next, the Demand Manager tries to identify the set of bids in response to customer RFQs that will maximize the expected profit from using the remaining production resources for the next 10 days. This profit depends not only on the RFQs being bid on on the current day, but also on RFQs that will be received on later days for computers due during the period. The Demand Manager therefore uses the future level of customer demand predicted by the Demand Model to generate a predicted set of all RFQs that will be received for computers due during the period. Bids for these RFQs are chosen at the same time as those for the actual RFQs from the current day, effectively causing a portion of the remaining production resources to be reserved for the actual RFQs that will be received in the future.

Once the predicted RFQs are generated, the Offer Acceptance Predictor is used to generate an acceptance prediction function for every RFQ, both real and predicted. The Demand Manager then considers the production resources remaining, set of RFQs, and set of acceptance prediction functions and simultaneously generates a set of bids on RFQs and a production schedule that produces the expected resulting orders. This process involves the use of a variation of our greedy production scheduler in which expected order quantities (where the expected quantity ordered for an RFQ is the probability of acceptance times the actual quantity requested) are considered. Full details are available in [6].

After applying the production scheduler to the current orders and RFQs, the Demand Manager is left with a 10-day production schedule and a set of bids for the actual and predicted RFQs. The bids on actual RFQs are sent to the customers, and the first day of the production schedule is sent to the factory specifying the instructions for the next day.

Finally, the Demand Manager projects component use for the period from 11 to 40 days in the future by using the Demand Model to predict customer demand and assuming that some fraction of this demand will be produced.

5 The Supply Manager

The Supply Manager is responsible for purchasing components from suppliers based on the projection of future component use provided by the Demand Manager, and for informing the Demand Manager of expected component deliveries and replacement costs. In order to be effective, the Supply Manager must be able to predict future component availability and prices. The Supplier Model assists in these predictions.

5.1 Supplier Model

The Supplier Model keeps track of all information sent to and received from suppliers. This information is used to model the state of each supplier, allowing the Supplier Model to predict the price that a supplier will offer in response to an RFQ with a given quantity and due date.

Recall that the price offered in response to an RFQ requesting delivery on a given day is determined entirely by the fraction of the supplier's capacity that is committed through that day. As a result, the Supplier Model can compute this fraction from the price offered. With enough offers, the Supplier Model can form a reasonable estimate of the fraction of capacity committed by a supplier on any single day. For each supplier and supply line, the Supply Manager maintains an estimate of free capacity, and updates this estimate daily based on offers received. Using this estimate, the Supplier Model is able to make predictions on the price a supplier will offer for a particular RFQ.

5.2 Supply Manager

The Supply Manager's goal is to obtain the components that the Demand Manager projects it will use at the lowest possible cost. This process is divided into two steps: first the Supply Manager decides *what* components will need to be delivered, and then it decides *how* best to ensure the delivery of these components. These two steps are described below.

Deciding What to Order. The Supply Manager seeks to keep the inventory of each component above a certain threshold. This threshold is 800, or 400 in the case of CPUs, and decreases linearly to zero between days 195 and 215. Each day, the Supply Manager determines the exact deliveries that would be needed to maintain the threshold on each day in the future given current inventory, expected deliveries, and projected component use. The result is a list of needed deliveries that we will call *intended deliveries*. When informing the Demand Manager of the expected future component deliveries, the Supply Manager will add these intended deliveries to the actual deliveries expected from previously placed component orders. The idea is that although the Supply Manager has not yet placed the orders guaranteeing these deliveries, it intends to, and is willing to make a commitment to the Demand Manager to have these components available.

Deciding How to Order. Once the Supply Manager has determined the intended deliveries, it must decide how to ensure their delivery at the lowest possible cost. We simplify this task by requiring that for each component and day, that day's intended delivery will be supplied by a single order with that day as the due date. Thus, the only decisions left for the Supply Manager are when to send the RFQ and which supplier to send it to. For each individual intended delivery, the Supply Manager predicts whether sending the RFQ immediately will result a lower offered price than waiting for some future day, and sends the RFQ if this is the case.

In order to make this prediction correctly, the Supply Manager would need to know the prices that would be offered by a supplier on any future day. Although this information is clearly not available, the Supplier Model does have the ability to predict the prices that would be offered by a supplier for any RFQ sent on the current day. To enable the Supply Manager to extend these predictions into the future, we make the simplifying assumption that the price pattern predicted on the current day will remain the same on all future days. This assumption is not entirely unrealistic due to the fact that agents tend to order components a certain number of days in advance, and this number generally changes slowly. Essentially, the Supply Manager follows a heuristic saying, "Given the current ordering pattern of other agents, prices are lowest when RFQs are sent x days in advance of the due date, so plan to send all RFQs x days in advance."

The final step is to predict the replacement cost of each component. The Supply Manager assumes that any need for additional components that results from the decisions of the Demand Manager will be felt on the first day on which components are currently needed, i.e., the day with the first intended delivery. Therefore, for each component's replacement cost, the Supply Manager uses the predicted price of the first intended delivery of that component, even if no RFQ was sent.

6 Adaptation over a Series of Games

The predictions made by the predictive modules as described above are based only on observations from the current game. Another source of information that could be useful in making predictions is the events of past games, made available in log files kept by the game server. During the final rounds of the TAC SCM competition, agents are divided into brackets of six and play a number of games (16 on the final day of competition) against the same set of opponents. When facing the same opponents repeatedly, it makes sense to consider adapting predictions in response to completed games. TacTex-05 makes use of information from these games in its decisions during two phases of the game: buying components at the beginning of the game (impacting mainly the behavior described in

Section 5.2), and selling computers at the end of the game (impacting the behavior in Section 4.2). We chose to focus on these areas for two reasons. Behavior during these two phases varies significantly from one agent to another, possibly due to the fact that these phases are difficult to reason about in general and may thus be handled using special-case heuristic strategies by many agents. At the same time, each agent's behavior remains somewhat consistent from game to game (e.g. many agents order the same components at the beginning of each game). This fact is critical to the success of an adaptive strategy – the limited number of games played means that it must be possible to learn an effective response from only a few past games.

6.1 Initial Component Orders

At the beginning of each game, many agents place relatively large component orders (when compared to the rest of the game) to ensure that they will be able to produce computers during the early part of the game. Prices for some components may also be lower on the first day than they will be afterwards, depending on the due date requested. Determining the optimal initial orders to place is difficult, because no information is made available on the first day of the game, and prices depend heavily on the orders of other agents.

TacTex-05 addresses this issue by analyzing component costs from past games and deciding what components need to be requested on the first two days in order to ensure a sufficient supply of components early in the game and to take advantage of low prices. The process is very similar to the one described in Section 5.2, except that predictions of prices offered by suppliers are based on past games. First, the components needed are identified, then the decision of which components should be requested is made, and finally the RFQs are generated.

The Supply Manager begins by deciding what components will be needed. On the first day, when no demand information is available (customers begin sending RFQs on the second day), the Supply Manager assumes that it will be producing an equal number of each type of computer, and projects the components needed to sustain full factory utilization for 80 days. On the second day, the Supply Manager projects future customer demand as before and assumes it will receive orders for some fraction of RFQs over each of the next 80 days. The projected component use is converted into a list of intended deliveries as before.

Next, the Supply Manager must decide which components should be requested on the current day (the first or second day of the game). As in Section 5.2, the Supply Manager must determine which intended deliveries will be cheapest if they are requested immediately. At the beginning of the game, the Supplier Model will have no information to use in predicting prices, and so information from past games is used. By analyzing the log from a past game and modeling the state of each supplier, it is possible to determine the exact price that would have been offered in response to any possible RFQ. Predictions for the current game can be made by averaging the results from all past games. When modeling the states of suppliers, RFQs and orders from TacTex-05 are omitted to prevent the agent from trying to adapt to its own behavior. If the initial component purchasing strategies of opponents remain the same from game to game, these average values provide a reasonable means of estimating prices.

At the beginning of the game, the Supply Manager reads in a table from a file that gives the average price for each component for each pair of request date and due date. Using this table, the Supply Manager can determine which intended deliveries will cost less if requested on the current day than on any later day. Intended deliveries due within the first 20 days are always requested on the first day, however, to avoid the possibility that they will be unavailable later. If opponents request many components on the first day of the game but few on the second, the prices offered in response to RFQs sent on the second day will be about the same as if the RFQs had been sent on the first day. Since information about customer demand is available on the second day of the game but not the first, it might be beneficial to wait until the second day to send RFQs. For this reason, the Supply Manager will not send a request for an intended delivery if the price expected on the second day is less than 3% more than the price expected on the first.

Once the Supply Manager has decided which intended deliveries to request, it must decide how to combine these requests into the available number of RFQs (five, or ten if there are two suppliers). In Section 5.2, this problem did not arise, because there were typically few requests per day. On the first two days, it is possible for the number of intended deliveries requested to be much larger than the number of RFQs available. Intended deliveries will therefore need to be combined into groups, with delivery on the earliest group member's delivery date. The choice of grouping can have a large impact on the prices offered. When there is only one supplier, the Supply Manager begins by dividing the 80 day period into five intervals, defined by six interval endpoints, with a roughly equal number of intended deliveries in each interval. Each interval represents a group of intended deliveries that will have delivery requested on the first day of the interval. One at a time, each endpoint is adjusted to minimize the sum of expected prices plus storage costs for those components delivered early. When no more adjustments will reduce the cost, the Supply Manager sends the resulting RFQs. When there are two suppliers, ten intervals are used, and intervals alternate between suppliers.

6.2 Endgame Sales

Near the end of each game, some agents tend to run out of inventory and stop bidding on computers, while other agents tend to have surplus computers, possibly by design, that they attempt to sell up until the last possible day. As a result, computer prices on the last few days of the game are often either very high or very low. When end-game prices will be high, it can be beneficial to hold on to inventory so as to sell it at a premium during the last days. When prices will be low, the agent should deplete its inventory earlier in the game. TacTex-05 adapts in response to the behavior of its competitors in past games by adjusting the predictions of the Offer Acceptance Predictor (Section 4.2) during the last few days of each game.

TacTex-05's endgame strategy is essentially to reserve only as many computers for the final few days as it expects to be able to sell at high prices. In particular, from day 215 to 217, the Demand Manager will always respond to a customer RFQ (if it chooses to respond) by offering a price slightly below the reserve. For RFQs received on these days, the probability predicted by the Offer Acceptance Predictor is set to the fraction of computers that would have sold at the reserve price on that day in past games. When the Demand Manager plans for a period of production that includes one of these days, these acceptance probabilities will hopefully result in an appropriate number of computers being saved for these three days.

7 Competition Results and Additional Experiments

In this section, we look at the results of the final day of the 2005 TAC SCM competition to determine how TacTex-05's adaptivity contributed to its performance. We also present the results of controlled experiments designed to test this adaptivity under differing conditions.

Out of 32 teams that initially entered the competition, 24 advanced past a seeding round to participate in the finals, held over three days at IJCAI 2005. On each day of the finals, half of the teams were eliminated, until six remained for the final day. Game outcomes depended heavily on the six agents competing in each game, as illustrated by the progression of scores over the course of the competition, underscoring the potential value of adaptation. In the seeding round, TacTex-05 won with an average score of \$14.9 million, and several agents had scores above \$10 million. Making a profit was much more difficult on the final day of competition, however, and TacTex-05 won with an average score of only \$4.7 million, followed by SouthamptonSCM with \$1.6 million and Mertacor with \$.5 Million. The other three agents (each of which averaged at least \$6 million in the seeding round) lost money.[3]

In order to visualize the game results from the final day of competition, we tracked four quantities over the course of each game, and plotted the average over all 16 games, shown in Figure 4. These quantities are component costs, time between component order and use, revenue, and profit. To determine daily component costs for each agent, a record of each component order was placed into a queue at the time the order was delivered. Whenever a computer order was delivered to a customer, components were removed from the queue, and the cost (including storage costs) recorded for that day (the day of delivery). Revenue from computer deliveries was similarly recorded on the day of the delivery. Penalties were not tracked. The precise meanings of the quantities graphed in Figure 4, from top to bottom, are as follows. Cost represents the average cost, as a fraction of the base price, for each component used (delivered as part of a computer) on a given day. Order time represents the average length of time between the date a component is ordered and the date a component is used, regardless of when the component is delivered to the agent. Revenue represents

[3] Competition scores are available at `http://www.sics.se/tac/scmserver`

the total sales prices for all orders delivered on a given day. Profit is equal to revenue minus all costs for the day. Thus both costs and order times are given as an *average* for all components *used* (not ordered) on a given day, while revenues and profits represent *totals* for the day. The data shown in Figure 4 has been smoothed with a Gaussian filter with a standard deviation of 5 days to reduce day-to-day noise. For clarity, only the data for the top three agents is shown. Similar patterns can be observed in the data for the three remaining agents. Below, we discuss what can be learned from these graphs about the performance of TacTex-05's adaptive methods.

7.1 Initial Component Orders During the Competition

We begin by examining the component orders placed by TacTex-05 at the beginning of games. Recall that TacTex-05 will order components on the first or second day of a game if it predicts that prices will be lowest on these days, based on the results of past games. (For simplicity, we will say that components were ordered on a day when in fact the order was placed the following day in response to an offer.) Figure 2 shows the number of components ordered on the first and second day of each game. As two games were played at a time, there are eight points plotted per day, each representing the average of two simultaneous games. For the first pair of games, when no previous games against the same opponents were available to use in predicting prices, data from the previous day of the competition was used. After the first two games, price predictions were based only on games from the final day of competition. First day orders jumped immediately after the first pair of games, and continued to rise to about 95,000, while second day orders gradually dropped to nearly zero. By the later games, TacTex-05 was ordering nearly all of the components it expected to use over the first 80 game days on the very first day. For comparison, the second largest orders came from the agent Deep Maize, which averaged 40,000 first day and 7,000 second day orders. SouthamptonSCM averaged 22,000 first day and 6,000 second day orders, Mertacor averaged 18,000 first day and 10,000 second day orders, and the other two agents had somewhat smaller orders. While the order sizes of the agents other than TacTex-05 did vary from game to game, they did so by relatively small amounts, and it is not clear if such changes were related to intentional adaptation. The fact that TacTex-05's first day orders continued to increase suggests the possibility of a self-reinforcing process: when more components are ordered on the first day, prices on later days may be driven up, making first day orders even more appealing.

The effects of these large first day orders on TacTex-05's performance can be seen in Figure 4. The most significant of these graphs is the daily profit. For roughly the first 40 days, TacTex-05's profit is below the profits of the other two agents, but between days 40 and 90, TacTex-05's profit is much higher. After day 90, TacTex-05 and SouthamptonSCM have mostly similar profits.The differences in profits during these two periods can be explained as a result of the large first day orders.

During the period in which TacTex-05's profit is highest, it receives slightly more revenue than SouthamptonSCM, but not enough to explain the gap. The difference must therefore be in costs, and this is seen to be the case in the costs graph. This difference in costs comes as no surprise – TacTex-05 ordered most of its components on the first day precisely because it expected costs to increase on later days. What *is* a surprise is the fact that TacTex-05's costs are higher than those of other agents for the first 40 days. These higher costs (along with what the revenue graph suggests is a slight lag in initial computer deliveries) explain the lower profit during this period, but are unexpected given TacTex-05's attempts to place orders in a way that minimizes cost. The explanation is related to the limited number of RFQs allowed per supplier per day. After deciding which components need to be ordered on the first day, TacTex-05 must decide how to group these components into the allowed number of RFQs, as described previously. The cost per component for one group can only be reduced by increasing the costs of another group, and so the differences between the two periods can be seen as a tradeoff: lower prices during the second period at the expense of higher prices in the first. Because the other agents order fewer components on the first day, they are not as affected by this tradeoff, and can obtain lower prices during the early part of the game.

7.2 Experimenting with Initial Component Orders

To see how TacTex-05's early-game adaptation would perform under different circumstances, and to better measure the impact of this adaptation, we ran an experiment using two versions of TacTex-05. A non-adaptive version used the price predictions resulting from the final day of the competition, leading to initial orders similar to those in the later games of Figure 2. An adaptive version began with these same predictions, but then adapted them as normal. 30 games were played between these agents and four agents that were not part of the final day of competition: PhantAgent, Botticelli, RationalSCM, and CrocodileAgent.[4]

Figure 3 shows the changes in first and second day orders over the 30 games. For this set of opponents, it appears that prices are no longer consistently lowest on the first day. First day orders immediately drop to the lowest level allowed (since some components will always be ordered on the first day), and more components are usually ordered on the second day than the first. The variation in second day orders is mainly due to the fact that customer demand can be taken into account on the second day. Figure 5 shows that the adaptive agent had lower component costs during the first part of the game, and these lower costs resulted in much higher profits during the first 50 days. The reason for slightly higher profits and revenue for the non-adaptive agent shortly after day 50 is not immediately clear. The average score of the adaptive agent was $.81 million higher than the non-adaptive agent's score, and this difference is statistically significant with 95% confidence according to a paired t-test. The results of this experiment indicate that optimal first and second day orders may be very different for different sets of opponents, and that TacTex-05 is able to quickly adapt to take advantage of this fact.

[4] Taken from the TAC Agent Repository, `http://www.sics.se/tac/showagents.php`

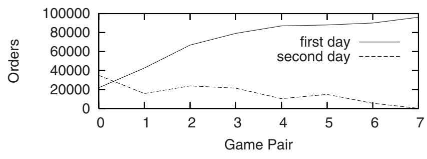


Fig. 2. Initial orders – competition

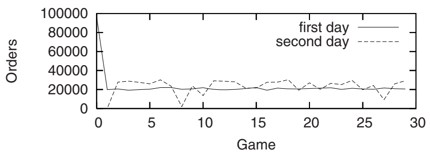


Fig. 3. Initial orders – experiment

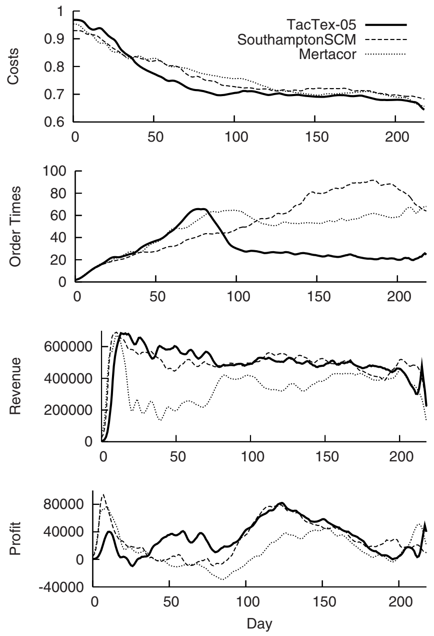


Fig. 4. Results – final day of competition

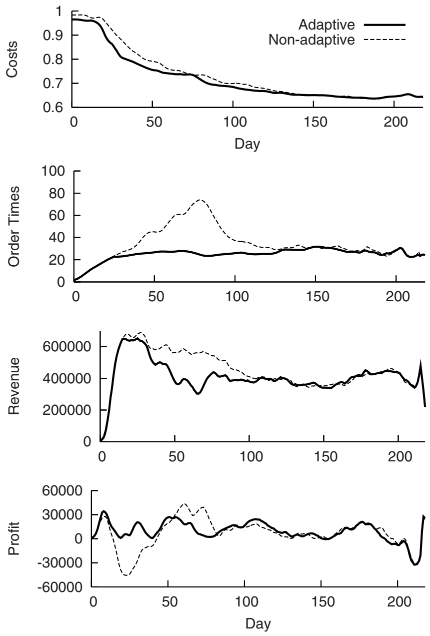


Fig. 5. Results – experiment

7.3 Endgame Sales During the Competition

The goal in adaptively predicting offer acceptance probabilities at the end of a game is to determine whether computers should be held back in hopes of high prices or sold early before prices drop. The first strategy turned out to be the correct one during the final day of the competition – on average, the fraction of customer RFQs receiving offers dropped from 57% on day 215 to 37% on day 217, and prices rose accordingly. From that standpoint, TacTex-05 behaved correctly, reducing computer sales near the end of the game and then increasing them during the last few days, as shown by Figure 4. However, TacTex-05 sold fewer computers overall during the last 20 days than it could have, and did not have particularly high profits during this period. The problem appears to be a lack of components caused by factors other than the adaptation, such as the reduction of the inventory threshold.

7.4 Experimenting with Endgame Sales

Since it is difficult to evaluate the success of the endgame adaptation that took place during the competition, we instead rely on experimental evaluation. We performed an experiment using four different versions of TacTex-05. One used endgame adaptation as normal, one assumed 100% offer acceptance during the final three days, one assumed 0% acceptance, and one (which we'll call "standard") used the same method of predicting offer acceptance as during the rest of the game. The other two agents were the "dummy" agents provided by the game server. The results are presented in the first column of Table 1. The adaptive agent outperformed the 0% agent and the standard agent (statistically significant with 95% confidence according to paired t-tests), but was slightly worse than the 100% agent (not significant). The 0% agent and the standard agent both had fairly large penalties during the final few days that accounted for much of the difference, caused by the fact that they underestimated offer acceptance probabilities.

Although the 100% agent worked well, under different circumstances this strategy might not work. To demonstrate, we ran another experiment with one adaptive agent and three 100% agents. The results are presented in the second column of Table 1. This time, the adaptive agent outscored the 100% agents (statistically significant). The 100% agents tended to have more unsold inventory than the adaptive agent, indicating that they overestimated their chances of selling computers during the final days. These two experiments demonstrate the need to handle sales differently during the end of a game than during the rest of the game, and the drawbacks of using a fixed strategy.

Table 1. Endgame adaptation experiments

Agent	*Exp. 1 Score*	*Exp. 2 Score*
adaptive	$7.14M	$6.85M
100%	$7.20M	$6.41M
0%	$6.46M	–
standard	$6.74M	–

8 Related Work

A number of agent descriptions for TAC SCM have been published presenting various approaches to the tasks faced by an agent. (See `http://tac.eecs.umich.edu/researchreport.html` for a collection of papers.) Although several agents make efforts to adapt to changing conditions *during* a single game (e.g. [8], [9]), methods of adaptation to a set of opponents over a series of games in TAC SCM have not been reported on to our knowledge. (Such adaptation has been used in the TAC Travel competition, however, both during a round of competition [10], and in response to hundreds of previous games [11].) While attention has been paid to the problem of early-game component procurement, much of it has focused on an unintended feature of the game rules (eliminated in the 2005 competition) that caused prices to *always* be lowest on the first day of the game [12].

9 Conclusion

In this paper we described the TacTex-05 agent, focusing on its ability to adapt over a series of games. The adaptation of component purchases during the beginning of a game was shown to have a positive impact on performance during the 2005 TAC SCM competition, and both start-game and end-game adaptation were experimentally shown to allow the agent to respond to various types of opponent behavior. Improving TacTex-05's predictive modules through additional forms of adaptation remains an important area for future work.

Acknowledgments

We would like to thank Jan Ulrich for contributing to the development of TacTex-05, the SICS team for developing the game server, and all teams that have contributed agents to the agent repository. This research was supported in part by NSF CAREER award IIS-0237699.

References

1. Fox, M.S., Chionglo, J.F., Barbuceanu, M.: The integrated supply chain management system. Internal Report, Dept. Industrial Engineering, University of Toronto, available from http://www.eil.utoronto.ca/public/iscm-intro.ps (1993)
2. Sadeh, N., Hildum, D., Kjenstad, D., Tseng, A.: Mascot: an agent-based architecture for coordinated mixed-initiative supply chain planning and scheduling. In Workshop on Agent-Based Decision Support in Managing the Internet-Enabled Supply-Chain, at Agents '99 (1999)
3. Chen, Y., Peng, Y., Finin, T., Labrou, Y., Cost, S.: A negotiation-based multiagent system for supply chain management. In Workshop on Agent-Based Decision Support in Managing the Internet-Enabled Supply-Chain, at Agents '99 (1999)
4. Pardoe, D., Stone, P.: Predictive planning for supply chain management. In: Sixteenth International Conference on Automated Planning and Scheduling. (2006)
5. Kiekintveld, C., Wellman, M., Singh, S., Estelle, J., Vorobeychik, Y., Soni, V., Rudary, M.: Distributed feedback control for decision making on supply chains. In: Fourteenth International Conference on Automated Planning and Scheduling. (2004)
6. Pardoe, D., Stone, P.: Bidding for customer orders in TAC SCM. In: AAMAS 2004 Workshop on Agent Mediated Electronic Commerce VI: Theories for and Engineering of Distributed Mechanisms and Systems. (2004)
7. Benisch, M., Greenwald, A., Grypari, I., Lederman, R., Naroditskiy, V., Tschantz, M.: Botticelli: A supply chain management agent. In: Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS). Volume 3. (2004) 1174–1181
8. Ketter, W., Collins, J., Gini, M., Gupta, A., Schrater, P.: Identifying and forecasting economic regimes in tac scm. In: IJCAI 2005 Workshop on Trading Agent Design and Analysis. (2005) 53–60

9. He, M., Rogers, A., David, E., Jennings, N.R.: Designing and evaluating an adaptive trading agent for supply chain management applications. In: IJCAI 2005 Workshop on Trading Agent Design and Analysis. (2005)
10. Stone, P., Littman, M.L., Singh, S., Kearns, M.: ATTac-2000: An adaptive autonomous bidding agent. Journal of Artificial Intelligence Research **15** (2001) 189–206
11. Stone, P., Schapire, R.E., Littman, M.L., Csirik, J.A., McAllester, D.: Decision-theoretic bidding based on learned density models in simultaneous, interacting auctions. Journal of Artificial Intelligence Research **19** (2003) 209–242
12. Kiekintveld, C., Vorobeychik, Y., Wellman, M.P.: An analysis of the 2004 supply chain management trading agent competition. In: IJCAI 2005 Workshop on Trading Agent Design and Analysis. (2005)

Market Efficiency, Sales Competition, and the Bullwhip Effect in the TAC SCM Tournaments

Patrick R. Jordan, Christopher Kiekintveld, Jason Miller, and Michael P. Wellman

University of Michigan
Computer Science & Engineering
Ann Arbor, MI 48109-2121, USA
{prjordan,ckiekint,mijason,wellman}@umich.edu

Abstract. The TAC SCM tournament is moving into its fourth year. In an effort to track agent progress, we present a benchmark market efficiency comparison for the tournament, in addition to prior measures of agent competency through customer bidding. Using these benchmarks we find statistically significant increases in intratournament market efficiency, whereas agents are generally decreasing in manufacturer market power. We find that agent market share and bid efficiency have increased while the variance of average sales prices has been significantly reduced. Additionally, we test for a statistical relationship between agent profits and the bullwhip effect.

1 Introduction

The supply chain management (SCM) game of the Trading Agent Competition (TAC) has provided three years of rich competition among a diverse pool of participants. We seek to evaluate progress and changes in the field of agents, and employ a variety of measures for this evaluation. These include measures of both social welfare and individual performance. We also raise the issue of bullwhip effects in the TAC SCM game, since this is a commonly discussed phenomenon in other supply chain settings. Our analysis is complicated by the strategic interactions that played out in the component procurement markets on day 0 during the early years of competition, and the subsequent changes to the game specification. We will discuss the possible effects of these changes at relevant points in the discussion.

In Section 3 we discuss a method for calculating market efficiency and the division of surplus in the SCM game. Using this measure of efficiency as a benchmark, we compare agent performance between rounds in each of the three tournaments and note interesting trends between tournaments. Section 4 gives results comparing several measures of sales performance in the customer market. Since the specification of the customer market has changed less drastically than the supply market, comparisons across tournaments on these metrics hold more weight. In Section 5 we apply a basic measure of the bullwhip effect to the SCM market and consider the relationship of this measure with market efficiency and the division of surplus. We conclude with a summary of the results and discussion of the usefulness of the various measures.

2 TAC SCM Game

The TAC supply chain management game ([1], [2], [3]) consists of six manufacturing agents competing simultaneously in two separate markets over a period of 220 simulated days to assemble and sell PCs to customers. The manufacturing agents attempt to procure processors, motherboards, memory, and hard drives from eight suppliers at low cost, while assembling and selling 16 different types of PCs to customers at a high price. Each agent is assigned an identical factory for production. Factories have a limited number of cycles each day for production. PC types vary in the number of cycles necessary for production.

Supplier prices are determined by available capacity and inventory levels. The available capacity is driven by a mean reverting random walk, while inventory is determined by past excess capacity and orders already on the books. In 2005, a reputation component was added to the supplier pricing equation. An agent's reputation determines the order in which the agent's request for quotes will be considered. Lower reputations result in higher price offers. The agent's reputation is assigned by considering the ratio of requested amounts to actual amounts purchased. If that ratio is within some acceptable range (acceptable purchase ratio) the agent is designated as having a perfect reputation.

Daily customer demand is Poisson distributed about a mean for each market segment. The mean evolves according to a random walk with an evolving trend parameter. Each customer request for quote contains the requested PC type, quantity, due date, reserve price, and penalty amount. An agent receives an order if the agent issues the lowest bid that meets the reserve price. Agents pay the assigned daily penalty if they faily to deliver by the due date.

3 Market Efficiency and Power

In order to gauge the aggregate agent behavior in the TAC SCM market, we present market power and efficiency benchmarks for evaluating and analyzing the entire economy. We generally expect market efficiency to increase during each subsequent tournament. However, as noted in [4], market efficiency is not a direct measure of agent progress and competent agents may not necessarily maximize market efficiency. We also show the results of analyzing market power (distribution of surplus). The market power held by manufacturers tends to decrease in subsequent tournaments and across rounds.

3.1 Calculating Market Efficiency

We calculate market efficiency by comparing TAC SCM market revenue to the revenue generated by a central planner using complete knowledge of supplier capacity and customer demand. We assume that the supplier value of components is the discounted base price, $P_c^{base}(1-\delta)$,[1] and that the customer value of a PC in an order is the reserve price of the request for quote. Therefore the revenue extracted from the market for a single PC is the customer value minus the suppliers' cost of components, penalties, and storage costs. On any day d, a PC given in a customer RFQ may be produced from the available supplier components of days $2 \dots d-3$, denoted C_d, and the available factory capacity on days $3 \dots d-2$, denoted F_d. Therefore the constraints on a day d accepted RFQ set A_d are

i. The set of components $Components(A_d)$ required by A_d must be contained in the available component set C_d:

$$Components(A_d) \subseteq C_d(A_4, \dots, A_{d-1}).$$

ii. The factory cycles $Factory(A_d)$ required to produce A_d must be contained in the available factory cycles set F_d:

$$Factory(A_d) \subseteq F_d(A_4, \dots, A_{d-1}).$$

Therefore, we would maximize *social welfare* by finding the RFQ acceptance set A which solves the following problem

$$\begin{aligned} &\max_{\{A_d \;:\; A_d \subseteq RFQ_d\}} \sum_{d \in Days} \mathcal{V}(A_d) - \mathcal{C}(A_d) \\ &s.t. \quad \forall d \in Days \qquad Components(A_d) \subseteq C_d \\ &\qquad\qquad\qquad\qquad\quad Factory(A_d) \subseteq F_d \end{aligned} \tag{1}$$

where $\mathcal{V}(\cdot)$ is the customer value and $\mathcal{C}(\cdot)$ is the supplier component cost plus any storage costs and penalties. This amounts to a large mixed integer linear programming problem. For any given day, the *central planner* may use the current day's component availability and factory capacity as well as the remaining aggregate available components and factory capacity. In doing this we actually allow component availability and factory capacity to be scheduled for early production in order to fill orders that otherwise would not have been obtainable. It is typical behavior for agents in the tournament to procure components in the expectation of customer sales. In our implementation we make a simplification that allows for the possibly infeasible case where aggregate factory capacity is used to build PCs from aggregate component availability that is not yet technically available. For example, consider an acceptance set decision for day 30. An infeasible production state would occur if day 10 had available factory capacity that was used to manufacture PCs with components available on day 20 due for delivery on day 30. As mentioned later in the results regarding the *naive central planner*, shifting component availability and factor capacity accounts for less

[1] $P_c^{base}(1-\delta)$ is the lower bound on component price given in [3].

than 9% of market surplus. Given the current (2005) settings, factory capacity is usually the binding constraint and therefore we feel justified in considering the possibility and additionally the effect of generating infeasible production states negligible. A typical example of a game in the TAC 2005 finals contains over 35,000 values (potential orders) and 2,500 constraints (aggregate component availability and factor capacity), which is not computationally feasible for a standard solver. In order to approximate an optimal central planner, we create a *greedy central planner* that seeks to maximize Equation 1 given a subset of the game's potential orders subject to the constraints. For instance, suppose that we make a greedy choice for every simulation day, the central planner solves 220 instances of the 0-1 multiple knapsack problem in approximately 100 variables and 11 constraints.

3.2 0-1 Multiple Knapsack Solver

We test two different types of solvers for the 0-1 multiple knapsack problem (MKP). Both use a genetic algorithm to search for optimal solutions. The first solver, which we denote the direct search genetic algorithm (DSGA), searches the variable domain directly. The second is a variant of the hybrid genetic algorithm developed in [5], which we denote the meta-search genetic algorithm (MSGA). The MSGA uses a construction heuristic to search over perturbed problem spaces. Both genetic algorithms are tested using a population size of 100 with mutation rate $1/n$ where n is the length of the chromosome, two point crossover, and tournament selection with tournament size 4.

In DSGA, the objective function we use allows infeasible assignments. When a constraint is violated a set of *active* objects that contribute to the constraint are ordered by value. The *active* object with the minimum profit is deactivated and its value subtracted from the objective function. This process continues until all constraints are satisfied. The fitness of the remaining object set is then computed according to Equation 1. This treatment of infeasible strings is similar to [6] but differs in that the penalty term is added per constraint violation, which amounts to a smaller jump in fitness. We initialize the population by generating object sets such that each set is either infeasible or exactly satisfies some constraint. Each chromosome is created in the initial population by starting with all objects *inactive* and then sequentially activating until the chromosome is infeasible, exactly satisfies at some constraint, or is fully active.

We chose to compare DSGA and MSGA on a subset of the problems used by [5]. This also served as a validation of our implementation of MSGA. While DSGA was competitive with MSGA, MSGA did have a higher mean score in every problem instance. We chose to use our implementation of DSGA for the greedy central planner due to its significant superiority in execution speed.[2]

[2] On both the Java VM implementations for OS X and FreeBSD DSGA ran an order of magnitude faster than MSGA.

3.3 Market Efficiency Comparison

We consider changes in market efficiency both between years and rounds within the same tournament. These results are shown in Table 1, while the p-values for intratournament comparison are shown in Table 2. The comparisons between tournaments are presented with the caveat that specification changes prevent us from making definative judgements.

In 2005 there was a statistically significant improvement from semifinal to finals play while the improvement in agent efficiency from quarterfinals to semifinals was statistically insignificant. This is consistent with the understanding that teams are improving agents across rounds, and weaker agents are eliminated. This was quite different from the scenario in 2004. While agents exhibited a significant improvement in the quarterfinals to semifinals, this trend did not continue into the finals. Analysis by [7] identified the blocking strategy employed by *FreeAgent*, which barred agents from procuring non-CPU components for a substantial part of the game. This behavior, while competitive, seems to have resulted in an overall loss in market efficiency.

Table 1. Intratournament Market Efficiency

	QF	SF	F
2003	57.0%	56.3%	60.1%
2004	64.6%	73.1%	54.3%
2005	84.1%	84.8%	87.7%

QF - Quarterfinals, SF - Semifinals, F - Finals

Table 2. p-value of Intratournament Market Efficiency Comparisons

	2003			2004			2005		
	QF	SF	F	QF	SF	F	QF	SF	F
QF	1	0.42	0.13	1	2.8e-3	9.6e-4	1	0.21	2.8e-5
SF		1	0.12		1	1.8e-9		1	1.9e-3
F			1			1			1

In order to better understand the measure of market efficiency we tested two benchmarks which give a relative comparison for the TAC SCM 2005 tournament. The first benchmark used a set of six *Dummy* agents provided in the SCM server software. Given the 2005 configuration, this set of agents had a mean market efficiency of 54.4%. The second benchmark considers only the factory and component capacity for a given day, instead of the aggregate prior respective values, as the constraints on production. Using this approach the *naive central planner* achieves an average market efficiency of 91.2%.

3.4 Market Power Comparison

Given the supplier and customer utility defined above we analyze the distribution of surplus within the TAC SCM market over the prior years' tournaments. The market power distribution is given in Table 3 with the corresponding significances given in Table 4. We expect that a more competent field of agents will have lower aggregate market power. The 2004 tournament has a statistically significant decrease in market power from the semifinal to final rounds, while the 2005 tournament shows a decrease in the transition from quarterfinals to semifinals.

Table 3. Tournament Market Power Distribution

	QF			SF			F		
Year	S	M	C	S	M	C	S	M	C
2003	14%	39%	47%	22%	23%	55%	28%	11%	61%
2004	21%	32%	47%	16%	33%	51%	38%	9%	53%
2005	36%	13%	51%	48%	2%	50%	46%	4%	50%

S - Supplier, M - Manufacturer, C - Customer

Table 4. p-value of Intratournament Manufacturer Market Power Comparisons

	2003			2004			2005		
	QF	SF	F	QF	SF	F	QF	SF	F
QF	1	0.26	0.04	1	0.34	1.5e-4	1	1.4e-4	2.2e-3
SF		1	0.31		1	1.5e-5		1	0.19
F			1			1			1

4 Customer Market

One of our goals is to develop useful metrics for evaluating agent play in addition to overall profits. In this section we focus specifically on the customer market, presenting three different benchmarks that provide information about agent interactions in this market. These are *market share*, *average selling price* (ASP), and *bid efficiency*. One motivation for specific analysis of the customer market is that the structure of this market has remained relatively constant, allowing for somewhat better intertournament comparisons. There were some changes after 2003, with modifications to the demand evolution process and separation of the market into three distinct segments. In addition, we note that changes in the supplier market can have an effect on customer market measures.

4.1 Market Share

Market share numbers are presented in Table 5. The most obvious feature is that unbid market share dropped dramatically in 2005. Figure 1 shows a breakdown of the total satisfied fraction of the market by simulation day, averaged over

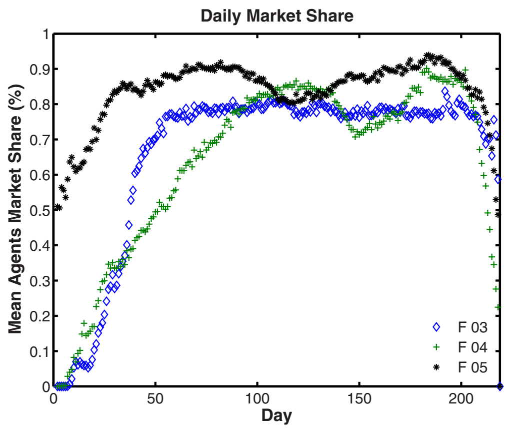


Fig. 1. Mean Daily Market Share

The figure shows the mean aggregate agent market share over the 2003, 2004, & 2005 finals

all finals games in the three tournaments. In 2003 and 2004 the fraction of the market satisfied early in the game is very low compared to 2005. The difference is attributable to changes in the day-0 purchasing behavior. In 2003 and 2004 the large orders placed on day 0 prevented agents from getting a full complement of components necessary for production until many days into the game in most cases. They were willing to make this tradeoff because of the very low prices for these components. Another explanation for the higher fraction of the market satisfied in 2005 is the increase in nominal supplier capacity from 500 to 550 components per line per day. This reduces prices and instances where production is not feasible due to supplier capacity constraints. Both of these are instances where changes to the supplier market had a clear effect on behaviors in the customer market.

Another interesting feature of the data may be less influenced by the changes in the supplier market. The variability in agent's market share decreased noticable in 2005, and particularly in the final round. This is an indicator of increased parity between the agents and possibly greater consistency in dealing with a variety of market conditions.

4.2 Average Sales Price

We now consider the average selling price (ASP) for PCs. All prices are normalized by the base price. We look particularly at the standard deviation of these values. In earlier tournaments, agents were able to corner markets for significant periods of time because of the strategic day-0 procurement issues. This resulted in a decreasing trend in customer ASPs over the course of the game, identified in [7]. While there is still a benefit to early production, this effect has been attenuated by the specification changes. This is noticable in the significant drops in the standard deviation of ASP in subsequent tournaments.

Table 5. Market Share per Agent

Year		Mean	Std. Dev.	Unbid
	QF	8.9%	2.8%	47.3%
2003	SF	8.3%	2.4%	50.2%
	F	9.6%	2.2%	42.2%
	QF	10.8%	1.2%	35.0%
2004	SF	10.1%	3.1%	39.4%
	F	10.1%	0.9%	39.7%
	QF	13.4%	1.1%	19.6%
2005	SF	13.6%	1.1%	18.4%
	F	13.5%	0.8%	19.0%

Table 6. TAC SCM Tournament ASP

	2003		2004		2005	
	Mean	Std. Dev.	Mean	Std. Dev.	Mean	Std. Dev.
QF	77.1%	30.1%	82.3%	21.2%	77.4%	15.4%
SF	76.2%	32.0%	80.2%	21.0%	78.3%	15.3%
F	78.2%	26.8%	84.0%	20.6%	78.5%	15.2%

4.3 Bid Efficiency

We define *bid efficiency* to be the ratio of actual revenue achieved by bids and the maximum possible revenue that could have been achieved by bidding on the same set of requests, but with perfect information about opponents' bids. In other words, this is the winning price divided by either the first losing bid or the reserve price if there was no other bid. This is an interesting measure because we can directly compare at least one aspect of agent behavior against the optimal behavior.

Table 7 shows the evolution of agents' bidding efficiency over the competition's three year history. All comparisons are statistically significant. Note that the standard deviation of agent bid efficiency in the 2005 tournament finals is less than 5.2% in comparison to 14.2% and 11.8% in the 2003 and 2004 tournament finals, respectively. One possible explanation for this is increased stability in the customer market prices, which would make it somewhat easier for agents to make bids close to the winning price levels. In general, increases in bidding efficiency reflect improvements in an agent's ability to predict the winning price levels.

Table 7. TAC SCM Tournament Bid Efficiency

	2003		2004		2005	
	Mean	Std. Dev.	Mean	Std. Dev.	Mean	Std. Dev.
QF	88.7%	15.6%	92.6%	11.0%	95.7%	6.7%
SF	87.9%	16.2%	93.8%	9.3%	97.1%	4.5%
F	89.0%	14.2%	94.0%	11.8%	97.1%	5.2%

Bidding efficiency is very high in the 2005 finals, so it is likely to be difficult to gain much advantage from improving this aspect of bidding performance in future agent designs.

5 Bullwhip Effect

The *bullwhip effect* is a widely known and commonly studied issue in supply chain scenarios [8]. The effect was made famous by the MIT beer game, which is often used to demonstrate the idea in business schools. The basic problem is that volatility in demand (orders) is amplified as it moves up the supply chain, making planning more difficult for entities further from end demand. The SCM scenario is a supply chain with three echelons: customers, manufacturers, and suppliers. Since it is a supply chain, we might expect that it would exhibit the bullwhip effect. We looked for evidence of this by comparing the relative demand variability in the Supplier-Manufacturer echelon and the Manufacturer-Customer echelon. For a specific supply web, we define the component demand C and the customer demand Q. An example of the supply web formed by component 100 and PCs 1, 2, 3, and 4 is shown in Figure 2. We can then quantify the bullwhip effect as the ratio of the component demand standard deviation to customer demand standard deviation[9]:

$$\omega = \frac{\sigma[C]}{\sigma[Q]} \tag{2}$$

The customer demand Q is Poisson distributed with an evolving trend parameter[3]. TAC agents must plan their component procurement according to this evolution in order to effectively manage the supply chain. Figure 3 shows these signals for the component 100 supply web of game 3717 in the 2005 finals. The bullwhip measure ω for this supply web is 1.37. Any measure greater than 1 indicates a possible instance of the bullwhip effect.

We measured the bullwhip effect for all of the 2005 finals games. The mean and standard deviation of ω for the finals are 2.35 and 0.36, respectively. In addition, we measured the correlation between Q and C, denoted $\rho_{Q,C}$. For the finals the mean and standard deviation for this correlation are 0.19 and 0.09, respectively. We performed linear regressions of the base bullwhip metric ω and the correlation measure against market efficiency and the division of surplus. The

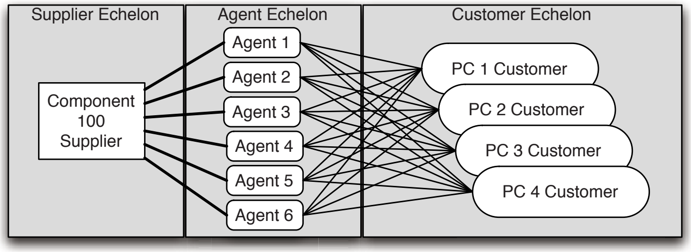


Fig. 2. Component 100 Processor Supply Web

results are shown in Table 8. The bullwhip measure had very little predictive power on any of the market measures, though there seems to be some relationship with the correlation measure.

We also considered bullwhip effects for individual agents. Figure 4 gives an example of the evolution of the C and Q signals for the six agents during a game. We regressed each agent's demand adjusted profits (DAP, calculated similarly to [10]) against ω and $\rho_{Q,C}$. The R^2 value for the ω regression is 3.8% and the R^2 value for the $\rho_{Q,C}$ regression is 17.1%, both with positive coefficients. The ω measure has very little explanatory power for agent scores, but well correlated procurement behavior may have positive benefits.

Table 8. Bullwhip Regression R^2

	ω Coefficient	ω R^2	$\rho_{Q,C}$ Coefficient	$\rho_{Q,C}$ R^2
Market Efficiency	0.086	1.7%	-0.089	13%
Supplier Power	0.014	1.47%	-0.199	21%
Manufacturer Power	0.017	0.8%	-0.215	8.4%
Customer Power	-0.03	2.2%	0.414	27%

Table 9. Finals' Agent Bullwhip

Agent	Mean	Std. Dev.
Deep Maize	2.69	0.24
TacTex	2.88	0.53
MinneTAC	1.41	0.16
Southampton	2.30	0.32
Maxon	2.52	0.77
Mertacor	1.23	0.15

In general, our current measures of the bullwhip effect do not seem to yield useful insights in the SCM domain. This does not mean that the bullwhip effect is not relevant. It may well be that our simple measures are not adequate to capture the complex dynamics of the SCM environment. Another issue is that the upstream agents (suppliers agents) in the SCM game are not very adaptive to changing demand conditions. This may be one reason that bullwhip effects are muted, particularly in how they effect overall measures of market efficiency. However, suppliers do have some behavior modifications based on demand, notably the behavior of producing ahead for future committments with unused capacity. This should be sufficient to see effects from better demand visibility under certain circumstances. We believe that the bullwhip effect is an interesting area for further investigation in the TAC/SCM game, and may shed new light on existing supply chain literature.

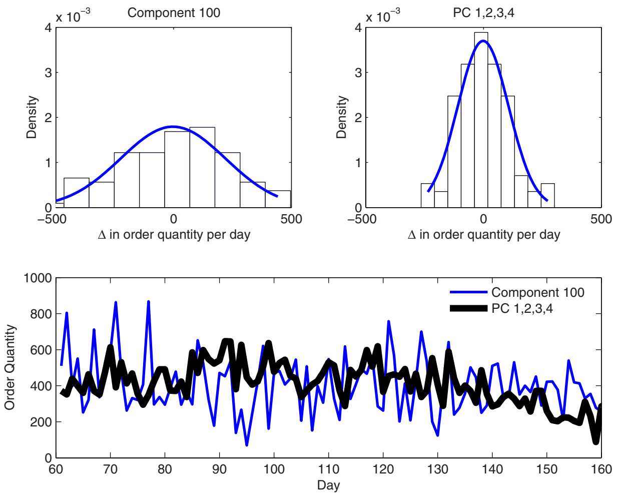


Fig. 3. Supply web signal for component 100 and PCs 1,2,3, and 4 in game 3717

The top-left plot shows the distribution of the change in order quantity per day of component 100. Similarly, the top-right plot shows the distribution of the change in order quantity per day of the PCs corresponding to component 100. These distributions are formed from the component and PC signals shown in the lower plot. Notice that the component, which is in a higher echelon, has a distribution with a significantly larger variance, signifying that there is a *bullwhip effect* in this supply web.

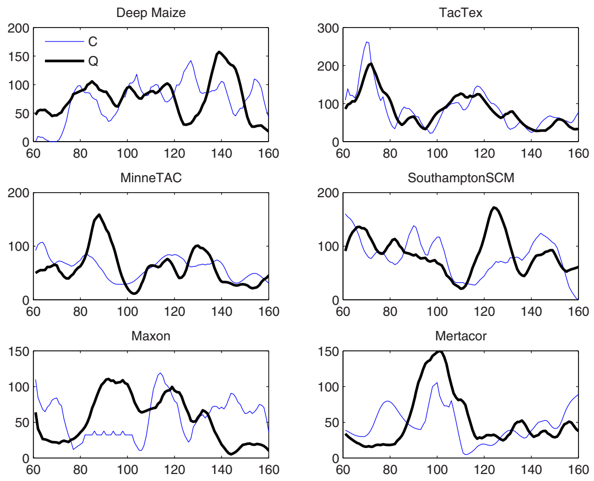


Fig. 4. Agent breakdown of supply web signal

6 Conclusion

We have examined a variety of measures of agent performance and overall market efficiency in the TAC SCM tournaments. These measures provide indirect evidence that agent competency is increasing, but it is difficult to separate out the effects of the changes in the game specification. The effects of these changes are pronounced in some of the measures. Overall market efficiency has increased in successive tournaments, while the market power held by manufacturing agents has decreased markedly. This is consistent with increases in the competitiveness of these agents. It is also an indication that the specification changes have increased competitive behavior. In the customer market, agents have increased market share while simultaneously increasing their bid efficiency. The variance on both bid efficiency and mean average selling prices has also decreased, effectively compressing the spread of relevant prices in the customer market. The addition of a repository of binary agents and stability in the specification will allow for more direct and extensive comparisons of agent performance in future tournaments. We also believe that there is a great deal to be learned by exploring widely studied phenomena like the bullwhip effect in the context of TAC SCM, and we look forward to further refinements of this analysis.

Acknowledgments

We would like to thank the TAC SCM participants for providing a rich, complex field of agents on which to perform analysis. We would also like to thank Kevin O'Malley for his involvement in design of our agent **Deep Maize**. This research was supported in part by the U.S. National Science Foundation under IGERT grant 0114368 and grant IIS-0205435.

References

1. Arunachalam, R., Sadeh, N.M.: The supply chain trading agent competition. Electronic Commerce Research and Applications **4** (2005) 63–81
2. Arunachalam, R., Eriksson, J., Finne, N., Janson, S., Sadeh, N.: The Supply Chain Management Game for the Trading Agent Competition 2004. Technical Report T2004-09, Swedish Institute of Computer Science (2004)
3. Collins, J., Arunachalam, R., Sadeh, N., Eriksson, J., Finne, N., Janson, S.: The Supply Chain Management Game for the 2005 Trading Agent Competition. CMU-ISRI-04-139 (2004)
4. Wellman, M.P., Chen, S.F., Reeves, D.M., Lochner, K.M.: Trading Agents Competing: Performance, Progress, and Market Effectiveness. IEEE Intelligent Systems **18**(6) (2003) 48–53
5. Cotta, C., Troya, J.: A Hybrid Genetic Algorithm for the 0-1 Multiple Knapsack Problem. In Smith, G., Steele, N., Albrecht, R., eds.: Artificial Neural Nets and Genetic Algorithms 3, Wien New York, Springer-Verlag (1998) 251–255

6. Khuri, S., Bäck, T., Heitkötter, J.: The Zero/One Multiple Knapsack Problem and Genetic Algorithms. In Deaton, E., Oppenheim, D., Urban, J., Berghel, H., eds.: Proc. of the 1994 ACM Symposium of Applied Computation proceedings, ACM Press (1994) 188–193
7. Kiekintveld, C., Vorobeychik, Y., Wellman, M.P.: An Analysis of the 2004 Supply Chain Management Trading Agent Competition. In Han La Poutré, N.S., Janson, S., eds.: Agent-Mediated Electronic Commerce: Designing Trading Agents and Mechanisms. LNAI 3937, Springer-Verlag (2006) 99–112
8. Lee, H.L., Padmanabhan, V., Whang, S.: The Bullwhip Effect in Supply Chains. Sloan Management Review **38**(3) (1997) 93–102
9. Fransoo, J.C., Wouter, M.J.: Measuring the bullwhip effect in the supply chain. Supply Chain Management **5**(2) (2000) 78–89
10. Wellman, M.P., Estelle, J., Singh, S., Vorobeychik, Y., Kiekintveld, C., Soni, V.: Strategic Interactions in a Supply Chain Game. Computational Intelligence **21**(1) (2005) 1–26

Agent Compatibility and Coalition Formation: Investigating Two Interacting Negotiation Strategies

Carlos Merida-Campos and Steven Willmott

Universitat Politècnica de Catalunya, Software Department, E-08034 Barcelona, Spain
{dmerida,steve}@lsi.upc.edu

Abstract. This paper focuses on the Coalition Formation paradigm as a market mechanism. Concretely, Coalition Formation occurs as part of a wider open world and may occur many times during the lifetime of a population of agents. This fact can in some circumstances be exploited by agents to re-use existing partial coalition and social relationships over time to improve Coalition Formation efficiency. The aim of the work is to analyze the dynamics of two concrete rational behaviors (Competitive and Conservative strategies) and, in particular, to investigate how agents in a heterogeneous population cluster together across multiple Coalition Formation episodes and varying tasks. Preliminary resuls are also shown regarding the manner in which playing distinct strategies interact with one another.

1 Introduction

One branch of economics research investigates how specific market situations force certain behaviors in their participants. A common example of this phenomenon is *Smith's Invisible Hand*, under which, in competitive market scenarios, traders are forced to set the prices where the supply meets the demand [13]. This specific situation does not appear too often in its pure form, and traders must often choose in what could be a large space of different pricing strategies.

The majority of agent-based research in the Economics area, focuses on studying strategy interactions and dynamics given a certain market. In this paper we focus on the related question of strategies and dynamics in market mechanism scenarios where Coalition Formation is required. Coalition Formation is a problem which has been extensively studied from Multi-Agent Systems perspective. The problem centers on finding subsets of agents from a general population to form groups which can most effectively carry out a particular task. The area includes research on a variety of different fronts, including problems such as finding core allocations of payoff between the members of a coalition [1], finding stability properties between coalition structures [7], finding solutions to a problem in a cooperative way [5], or finding solutions to a coalition problem in a competitive way, where coalitions compete amongst themselves for a payoff [12].

In the market mechanism we suggest here, coalition formation occurs as part of a wider open world and may occur many times during the lifetime of a population of agents. As shown in [12] this fact can in some circumstances be exploited by agents to re-use partial coalition and social relationships over time to improve Coalition Formation efficiency. Such a broader perspective however raises interesting questions for coalition formation environments. This paper goes further in analyzing the dynamics of two concrete rational behaviors, a Competitive Strategy and a Conservative Strategy within such an environment. These strategies are common examples of rational behavior in economic markets. Whilst there are more complex strategies, these two already show significant interesting behavior.

This type of continuously running market environment requiring coalitions reflects a large number of real world scenarios such as bidding for construction projects, the formation of consortia for product development or strategic alliances in emerging markets. In this context we aim to answer the following questions:

- To what extent do specific strategies affect the type of coalitions formed?
- To what extent do specific strategies affect the stability properties of the system?
- How do different agents with different strategies affect each other when they interact?
- Which strategies benefit agents and the general population more in which situations?

The paper is structured as follows: Section 2 explains the concrete market mechanism we are using, the Iterative RFP Coalition Formation method as well as examples to illustrate the method. Section 3 explains the agent based system designed to model the Iterative RFP Coalition Formation Method. Section 4 explains the characteristics of the strategies we are testing in this paper, and analyzes them from a theoretical perspective. Section 5 explains the experimental setup and the results obtained to underpin the theoretical analysis and to provide some new information. Section 6 covers related work and finally, Sections 7 and 8 provide conclusions and ongoing / future work repsecitvely.

2 Iterative RFP Coalition Formation Method

As the basis for experimentation, analysis in this paper adopts a model of worlds based on Request For Proposal (RFP from now on) scenarios. This model was first studied by [11], and further explored in [12]. In this environment, an entity or entities regularly issues a call for tender to provide specific goods or services with certain characteristics. Providers compete amongst themselves (either individually or in cosortia – *coalitions*). Providers and/or coalitions bidding for a particular call are ranked according to an evaluation of their skills for the task, and receive a payoff according to their placement in the ranking.

There are many existent real systems that follow the RFP type procedures such as public building projects, competitive tender for government contracts or even collaborative research project grants. RFP environments can also be seen as emerging market opportunities in an economy, with individual calls for tender representing new opportunities for profit. Such system are characterized as follows:

– Agents, or groups of agents compete for a given goal.
– The best agents, or groups of agents are rewarded with the award of the contract to carry out the task and its subsequently payoff.
– Some systems, may also have policies for rewarding 2nd, 3rd, 4th etc. ranked agents / groups of agents.
– The process repeats over time.
– The objective agents are competing for varies over time – with different CFPs issued corresponding to functionally different tasks. In this way, an agent or a group of agents that were very competent for a certain goal, could become weak for a different one.
– Groups could be dynamic and might change depending on the market situation.
– Agents are individual utility maximizers. And share the same preference, that is payoff maximization, but might have different strategies for maximizing their utility.

A protocol with the described characteristics could also potentially be used as a service composition mechanism in an automated environment where agents representing services team up to compete with other teams to present the most competitive package of services given certain requirements.

3 Problem Definition and Agent Based Models

For the purposes of this paper, the problem is formalized in the following way: In every game g there is a task T_g that is defined by a set of K tuples. Each tuple represents an skill and its corresponding demanding value for the named task:

$$T_g = \{\langle sk_0, T_{g0}\rangle, \langle sk_1, T_{g1}\rangle, \ldots, \langle sk_K, T_{gK}\rangle\}$$

Every agent A_i in the population has a certain expertise degree in each of the K skills that task T_g is defined with:

$$A_i = \{\langle sk_0, A_{i0}\rangle, \langle sk_1, A_{i1}\rangle, \ldots, \langle sk_K, A_{iK}\rangle\}$$

A coalition C_j is a a set of one or more agents $\{A_x, .., A_z\}$. Agents' skills are aggregated in the coalition in such a way that the resultant values of that aggregation represents the skills of the coalition entity. This is noted as:

$$C_j = \{\langle sk_0, C_{j0}\rangle, \langle sk_1, C_{j1}\rangle, \ldots, \langle sk_K, C_{jK}\rangle\}$$

The concrete aggregation function used is:

$$C_{jp} = \max_{\forall q: A_q \in C_j} A_{qp} \qquad \text{Having } 0 \leq p \leq K. \tag{1}$$

We could consider each skill as a necessary subtask for performing task T_g. In this way, by using the aggregation function shown in equation 1, the agent in a coalition which is the best fit for performing a certain subtask will be in charge of it.

Amongst the different possibilities for aggregating agent skills in a coalition, function 1 has been choosen as is a reasonable metaphor of many real coalitional processes. For example, if we consider a consortium of partners participating in a call for proposals, each member of the consortium will be representative of a certain part of the proposal, and normally is the partner best fitted for that part of the work.

Coalition C_j is endowed with a certain score $scr(C_j, T_g)$. This score is negative if the coalition is *non competent* in all the skills for performing the task ($\exists p : (0 \leq C_{jp} < T_{gp})$), and is positive otherwise. More concretely, the functions used for the case of existence of *non competent* skills in the coalition is:

$$scr(C_j, T_g) = -\#sk_p : (0 \leq C_{jp} < T_{gp}) \tag{2}$$

When Coalition C_j is competent in every skill ($\forall p : (0 < T_{gp} \leq C_{jp})$), the function used is:

$$scr(C_j, T_g) = \sum_{p=0}^{K} C_{jp} - T_{gp} \tag{3}$$

As we can see in Equation 2, the score of a coalition with some non competent skill (*non competent coalition*), is the negative value of the number of non competent skills. In this way, the more skills in which the coalition is not competent, the lower its score will be.

In Equation 3, we can see that the score of a coalition, competent in every skill (*competent coalition*) is the sum of excess value in every requested skill. In both equations, we can see that only those skills with value higher than 0 count for their evaluation. Those with values equal to 0 are ignored. This represents the fact that some tasks do not need a certain skill to be performed, and so the degree of ability of a coalition in that skill is not taken into account for its evaluation.

There is non-linear mapping from coalition score to coalition payoff, as in our model, the payoff of a coalition does not depend only on its score but also on the scores of other coalitions. All coalitions are decreasingly ordered by score, and are priced according to their rank with an exponentially decreasing amount from the best one to the worst. Agents within a coalition spread the coalition payoff evenly. The concrete payout function we use is:

$$\text{pay(rank)} = \begin{cases} \text{MaxAmount}/2^{\text{rank}-2}, & \text{for the last competent coalition} \\ \text{MaxAmount}/2^{\text{rank}-1}, & \text{for the other competent coalitions} \end{cases} \tag{4}$$

Amongst the different possibilities for doing the mapping, this concrete exponential function has been chosen for two reasons: first, because this function has the property that, independently of the number of competent coalitions, the total amount of money spread will always be $MaxAmount$. This is an interesting property in order to compare the economic behavior of different populations. The second reason is because this represents an exponential distribution of wealth for which there is empirical evidence from Real Economies (see [6]). This function creates a rich set of possible scenarios in which coalitions optimize their trade-offs between score and size, growing in size only when it is valuable to do so.

For every game, a subset of agents are asked at random about an action to take towards its membership in a coalition. The choices that an agent has are:

– Stay in the coalition.
– Stay in the coalition optimizing it by firing (expelling) one or more members.
– Leave the coalition in order to join a different one.
– Leave the coalition in order to replace one or more agents in a different one.
– Create a new coalition.

When an agent's decision involve firing or replacing an agent, this action is submitted to the coalition who will evaluate it. In order to get the action accepted and executed, it must be approved for more than the half of the members of the coalition affected, otherwise the action is rejected and not performed. In that case that an agent is fired or replaced, it automatically becomes the only member of a brand new coalition.

When agents are requested to perform an action they are able to submit as many proposals as they want, if none of them is accepted the agent remains in the same coalition.

Agents are farsighted in the sense that they know the score and payoff values of any action the agent wants to consider prior to its submission.

4 Conservative and Competitive Strategies

This paper shows results on the different outcomes obtained by using two different strategies: *competitive* and *conservative*. Agents who choose a *conservative* strategy make decisions on which coalition to join based on the payoff this coalition is expected to have. Agents who choose a *competitive* strategy make decisions on which coalition to join based on the score that this coalition is expected to have. Both strategies are myopically rational, as when an agent using some of those strategies is asked to make a choice, it counts with all the information available at that instant of time, concretely, they have the information on potential payoff and score of any coalition they could create by carrying out any of the 5 possible actions defined in the previous section, but they do not count on the possible reactions of the other agents after its decision has been performed.

Both strategies are arguably rational. For the case of *conservative* strategy, it makes sense to choose the most profitable coalition at a certain time, expecting that the situation will not change until the end of the game. For the case of *competitive* strategy, it makes sense to join the coalition with highest score, as even if the payoff is worse than in another coalition with lower score but fewer members to share the benefit, a new member could be attracted by this growing score coalition and make it grow in the ranking and gain a higher payoff. In other words, for the case of *conservative* strategy, the agent expects to get the maximum payoff at any time, while for *competitive* strategy, the agent invests for a future better payoff as a side effect of being in a highly competent coalition.

4.1 Theoretical Dynamics of Competitive Population

Competitive agents try to be in a coalition with the highest possible score. At the same time, coalitions of Competitive agents accept joining proposals, optimizations or replacements proposed as long as they improve the coalition score.[1] This behavior implies that, for a given task, each movement of an agent from/to a coalition of Competitive agents, involves an improvement in its score, otherwise the agent would not had had any motivation to move from/to there. Thus the score of the best coalition in a game for a given task is monotonically increasing.

The maximum score of a coalition is obtained with a coalition of as many members as skills required for the task (at maximum). Agents do not create coalitions with more members than skills, as given aggregation function 1, there is just one agent providing the maximum value for each skill, then if the coalition has more agents than required skills, it could be optimised by expelling those agents who do not provide any maximum value to any skill. Due to this, Competitive agents in the RFP model do not create the grand coalition, hence this strategy configures a non-superadditive game.

As the coalition size is self-limited by a certain maximum size, and the score of the best coalition is monotonically increasing, the score of the best coalition will stabilise at a certain point while the task doesn't change. The same happens with the score of the second best coalition, and so on. This way, a system with a population of Competitive agents competing for a given task that do not change, converges into an stable state.

4.2 Theoretical Dynamics of Pure Conservative Population

Conservative agents try to be in a coalition with the highest possible per agent payoff. At the same time, coalitions of Conservative agents accept joining proposals, optimizations or replacements proposed as long as they improve the coalition per agent score.[2] This behavior ensures that during the process of coalition formation, each movement of an agent from/to a coalition of Conservative agents, involves an improvement in the per agent payoff of the coalition otherwise, the agent would not have had any motivation to move from/to there. In order to improve the per-agent payoff, a coalition can either improve its score to increase its ranking, or it can reduce its size having less members to split the gains between less members. This second way of improving payoff implies in some cases a reduction of coalition score.

[1] As a secondary criterium of acceptance, we use the size of the coalition, i.e. if the coalition score is the same, a competitive agent will propose/accept when the number of agents is smaller than in the origin/original coalition.

[2] The same secondary criterion of acceptance as in competitive strategy is applied.

Differently to the best coalition's score, the best coalition's payoff is therefore not monotonically increasing. The best individual payment can decrease when the coalition that has it, is out-ranked by another coalition. In this case, the previous best payed coalition receives less payment as it is in a lower rank, and the out-ranking coalition, might have more members than the previous best-paying coalition, thus lower individual payoff. Under these conditions of non monotonicity, convergence to an stable state cannot be ensured in an environment with Conservative agents.

The sizes of coalitions are determined by the concrete skill distribution amongst the population, the requirements of the task and the payoff function. More concretely, the growing possibilities are determined by score differences from coalitions in the ranking. Let A_1 be an agent, C_1 be a coalition and $\backslash C_1$ the rest of coalitions competing at a certain moment. The ranking of C_1 is noted as $rank(C_1, \backslash C_1)$, its size as $|C_1|$, and its payoff as: $pay(rank(C_1, \backslash C_1))/|C_1|$. We can claim that C_1 will never grow in size as long as:

$$\nexists A_1 : pay(rank(C_1 \cup A_1))/(|C_1| + 1) > pay(rank(C_1))/|C_1| \tag{5}$$

Then the difficulty of a coalition has in growing depends on payoff function. Since the payoff function we are using in our model is monotonically decreasing as a function of the ranking of the coalition, in order to have a higher payoff in coalition $C_1 \cup A_1$, the following must be true:

$$rank(C_1 \cup A_1) > rank(C_1) \tag{6}$$

From this, we can see that the possibilities of a coalition to grow up in size also depend on how difficult what is stated in Condition 6 is.[3] As a matter of fact, the difficulty of condition 6 to hold for any arising coalitions, depends on the skill distribution of agents, and on the size of coalitions, in such a way that the smaller the coalition, the easier it is for condition 6 to be fulfilled. The Influence of coalition size is explained as follows: in a coalition with no redundant agents, each agent is giving the maximum value to one or more skills. If the coalition has few agents, its members will each have more skills to be responsible for. If agents of a population have the same total sum of skill values ($\sum_{p=0}^{K}(A_{ip})$), there will be many agents with different skill values in the distribution that could improve the coalition score by taking responsibility of many more skills than if the coalition were large and each skill would be responsibility of a certain "expert" agent. In this way, by having more chances of increasing coalition score by raising the value of more skills, we will have also more chances of rising the coalition rank.

The stability analysis in this population is more complex than in competitive population. The system will be stable once agents have reached Pareto optimality, and Nash equilibrium. This situation is made more complex by different

[3] Note that it could happen that joining more than one agent to the coalition ($C_1 \cup A_1 \cup \ldots A_n$) we could improve the coalition enough to make it profitable for all the agents in it, but given our model, agents do not take decisions in a coordinated manner.

factors. An important one is that if coalitions do not need to grow too much to be competent, we will have many small coalitions, and as we have seen in the analysis of the importance of the coalition size, for the case of small coalitions it will be easier for agents to find profitable coalitions outside their current coalition. This could create a continuous movement of agents from coalition to coalition.

5 Experiments

To examine the dynamics of the strategies explained and validate the theoretical analysis, a range of experiments were conducted by simulation. Each simulation run consists of a set of a fixed number of model iterations, where agents follow the Iterated *RFP* protocol explained in section 2, for solving a number of different tasks that change sequentially after a fixed number of games.

In order to visualize the relationships established between agents in the experiments, we used Pajek [3]. Graph figures represent the relationships created between agents throughout a series of games of one or more experiments. Each node is an agent, and a link represents a collaboration that existed when the coalition was evaluated in a certain game. Agents collaborate when they are together in a competent coalition. In this way, a coalition is represented by a clique of connections amongst the agents in the coalition. In order to ease the visual analysis edges are colored depending on the frequency of the relationship; the more often a collaboration happens, the darker the line appears. The graph is represented using *Kamada-Kawai* algorithm implemented in *Pajek* that places nodes in a close position when they are connected with links of relative high value. In our case, agents appear close to each other when they have had frequent relationships. Throughout the rest of the paper, these graphs are named *collaboration graphs.*

5.1 Experimental Set-Up

A variety of configurations and parameterizations have been used in order to be able to check the statistical validity of outcomes. The underlying configuration for the experiments was:

– A static population of 100 agents, with abilities randomly distributed across 10 different skills. Each skill of an agent is assigned a positive integer score or zero. Agents are each assigned 200 skill points randomly distributed across their skills.
– A set of 100 different Tasks. Each one requiring a total of 100 skill points distributed randomly across the same 10 different skills.
– Each task is issued during 1000 games. And each game picks 25 agents at random to make a choice on their coalition preference (and hence potentially adapt their coalition according to the schemes defined in the previous section).

– Every competent coalition is subsequently ranked and rewarded according to function 4. The concrete *MaxAmount* value used for this function is 100. This represents the total amount that will be spread in each game amongst the competent coalitions.

Collaboration Graphs are used to monitor each relationship established at the end of every round. The payoff data for each one of the agents, as well as the coalition sizes are also monitored and analysed.

5.2 Competitive Behavior Experiments

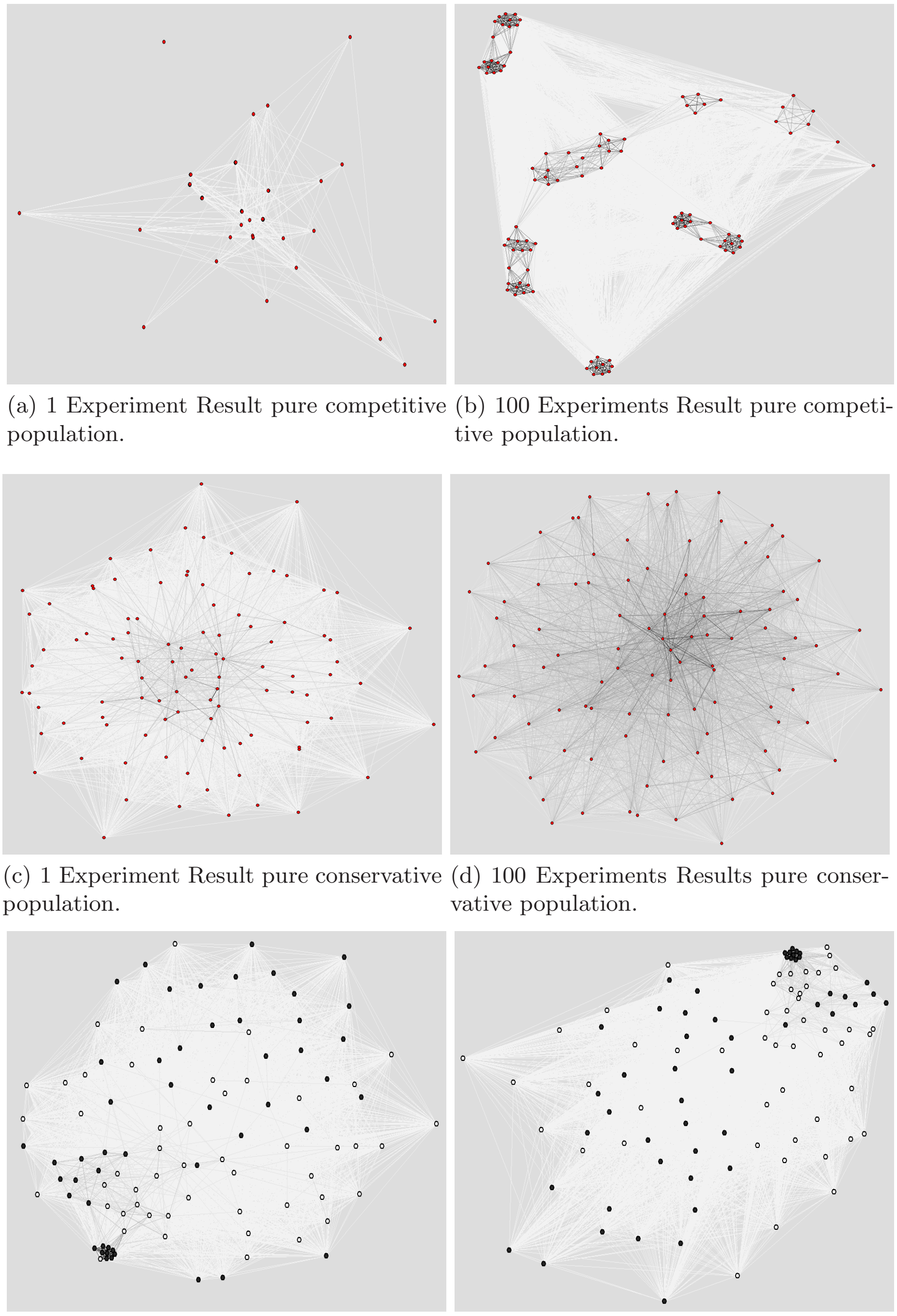
A population of 100 agents following the Competitive behavior strategy have been used. Figure 1(a) shows the *collaboration graph* of the population in just one experiment, and figure 1(b) represents the aggregated results amongst the set of 100 experiments with different tasks to fulfil. In the first figure we can appreciate 30 vertices out of the existing 100, this is because the algorithm places vertices close to each other when relationships between them are very strong (in our case, strength values means frequent relationships), that means that some coalitions of agents repeat very frequently over time. In the same figure, a few residual low frequency relationships of some agents with some of other clusters can be appreciated. Those two facts verify the analysis performed in section 4.1. Concretely, the experiments verify that a population of agents using Competitive strategy converges to a stable state where at a certain point of the process, there is no movement of agents between coalitions. Those clusters of nodes in the graph that repeat very frequently are the structures created after the optimization process has finished. Residual relationships (relationship with low frequency) belong to the period in which the experiment is not stable yet.

The experiments also reveal another fact that validates the theoretical analysis. The size of coalitions tends to be equal to the number of requested skills by the task, having normally an expert agent on charge of every skill.

The second figure shows us that even when different tasks are used in the same experiment, the aggregated collaboration graph still reveals clear clusters of frequent collaboration. This is explained by the following fact: Competitive agents create coalitions that maximize the value of all the skills requested by a task. In order to do that, they create coalitions as big as necessary, having an expert agent for each requested skill, i.e. an agent which only contribution is to have the biggest value in the coalition for an specific skill. The only difference in coalitions created for different tasks is in the exclusion or inclusion of expert agents for skills that in some tasks are requested and in others are not. This way a substantial part of a coalition remains static from task to task, as in average, tasks issued usually have no more than 2 non-required skills (that vary from task to task) out of 10 skill.

5.3 Conservative Behavior Experiments

In these experiments, we use a population of 100 agents following the conservative behavior strategy. Figure 1(c) shows the *collaboration graph* of the



(a) 1 Experiment Result pure competitive population.

(b) 100 Experiments Result pure competitive population.

(c) 1 Experiment Result pure conservative population.

(d) 100 Experiments Results pure conservative population.

(e) 1 Experiment Result mixed population.

(f) 100 Experiments Results mixed population.

Fig. 1. Collaboration Graphs for different populations settings

population of just one experiment, and figure 1(d) represents the aggregated results amongst the set of 100 experiments with different tasks to fulfil.

In the first figure we can appreciate how collaboration is not concentrated in clusters, instead it is spread in many different combinations of agents. This suggests the correctness of the analysis performed in section 4.2, where it was stated that a conservative population does not necessarily converge into an stable state. This way agents move from coalition to coalition during the experiment, and the collaboration graph reflects that behavior by not showing any particular frequent cluster of agents, instead, the graph shows a cloud of sparse collaborations between the agents. Conservative agents optimise the ratio payoff/members keeping low the size of coalitions. The experiments reflect an average size of 2.3. The small size of coalitions, is a destabilisation fact that make them have similar scores. When coalitions have similar scores a single movement of an agent can change the whole payment scenario. However, some nodes in the center of the graph show slightly more frequent connections between them. This suggests that even though the system does not converge, there are successful agents that have some preferential attachments with other nodes with which they can create small coalitions with good scores, and so, it happens that in a dynamically changing environment they meet each other more frequently.

By observing figure 1(d) representing the aggregation of the data obtained across all the experiments, we can see that it keeps the same concentric structure, indicating that successful properties of central agents are kept. By the dark color in the edges, we can see that although conservative agents spread their collaboration, they usually cooperate with the same wide range of agents.

5.4 Mixed Strategies Dynamics Experiments

In these experiments, we mix 50 agents using the competitive strategy with 50 agents using conservative strategy. In order to create comparable results when playing two populations together, we create a symmetric population of 50 duplicated skills vectors. Figure 1(e) shows the *collaboration graphs* of the population in just one experiment. Figure 1(f) on the right hand side represents the results

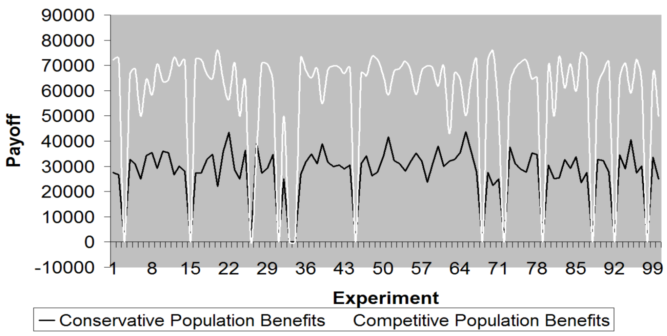


Fig. 2. Payoff gained by each population throughout 100 different tasks

amongst the set of 100 experiments with different tasks to fulfil. To differentiate members of each population in the graph, nodes representing conservative agents are in black color, and nodes representing competitive agents are in white color.

In the first graph we can see how a group with a majority of competitive agents establishes itself in the experiment, and show almost pure endogamic relationships. Apart from this, we observe that the rest of agents spread their collaboration with many different members of any population type. The average coalition size in this experiment is 5.9. This indicates that competitive agents boost the competitivity in the games, and large coalitions created by those agents become attractive also to conservative agents, as once the coalition has a certain competitive degree it becomes more profitable than small coalitions created by other conservative agents.

The figure showing results throughout the 100 experiments 1(f) reflects a very similar situation as in the one experiment graph 1(e). Both figures have a shape that reflects the mixture between clustering into frequent groups and spreading among many different ones. It is possible to appreciate a cluster of competitive agents, and very close to that cluster, reflecting frequent collaboration with the elements of the cluster, there is a set of conservative agents. Those agents do not stay in the cluster coalitions as frequently as competitive agents, because they are probably tempted by other coalitions offering less score and higher payoff, but they are probably good enough to be accepted in the coalitions when those coalitions become the most profitable option for a conservative agent at a certain time.

In terms of profit, as we can see in figure 2, the competitive population clearly outperforms conservative population. This is explained by the fact that competitive agents are the founder creators of the most competitive coalitions and remain on it as they cannot find any incentive to leave it. On the other hand, conservative agents that get attracted by a successful high score coalition might join it if its payout/members ratio is good enough, but they might eventually leave it seeking apparent opportunities in the market of smaller coalitions with higher benefits. This often turns out to be the wrong decision as smaller coalitions are more dynamic and subject to change (see section 4.2 for an explanation of this fact).

6 Related Work

There is other relevant work in literature that tackles heterogeneity of populations as a key point of the research. The most representative is [5] which studies how diversity within the agent population impacts on the quality of the coalitions that emerge. Conforth et. al. create a dynamic organization framework in which heterogeneity turns out to be a crucial element for problem solving tasks. In this work, heterogeneity is represented upon different initial values of agent's parameters such as their connectivity (interaction links), trading strategies and initialization states. In the present model, all agents know each other but are distinguished by different values in a fixed set of skills. This fact characterizes every individual by intrinsic properties that can be complementary, and lead to the system to have different dynamics from those showed in the cited work. Apart from using heterogeneity in the intrinsic properties of the agent, the presented model contemplates different strategic behavior for agents.

Another interesting example on the use of heterogeneity in coalition formation processes is the modelling of heterogeneous preferences. Some examples of this are [2] which models different preferences between leisure and work, and [4] which provides a method that considers the possibility of different evaluation functions to coalition structures.

From the point of view of the protocol used (RFP), the work presented is related to [11]. However, Kraus et. al. have radical differences in its use. Firstly, in their model, a number of tasks are issued and agents propose coalitions (from scratch) that are accepted or rejected by the affected members. In order to motivate agents to form a coalition, a discount factor is used. In the model presented here, there is just one Task at a time for which all the population compete for, but not only one coalition gets priced. Agents do not propose entire structures, instead they construct them by individual movements. In order to motivate agents to form good coalitions, the motivation instrument is the use of a decreasing payment function.

The presented model shares many characteristics with economic models, such as [2]. Some important characteristics are: the iterative nature of the processes, the episodical evaluation of the structures, and similar type of rational behavior in the agents, however there is an important difference that is the evaluation function. The evaluation function applied by Axtell et. al. (Cobb-Douglass) rewards a coalition (or a *firm*) independently of the rest of existing coalitions. In our case the reward of a coalition is partly dependant on its score, and on the score of the rest of the existent coalitions.

Traditionally Coalition Formation problems have been tackled as a "one off" event. In the present work we seek to go beyond this to consider what may happen in environments over time. Other important work in the same line includes [9,10,1].

7 Conclusions

From the analytical and experimental work presented, the following conclusions, applicable to the Iterative RFP domain, are drawn:

- Competitive strategies (score maximizing) outperforms conservative strategies (payoff maximizing) when symetric populations are played against one another. This provides a hint as to why competitivity emerges in certain societies to dominate other conservative behaviours.
- A pure population of agents running the competitive strategy for a give task, converges to a Nash equilibrium state in which no agent has motivation to move elsewhere given the coalitional structure created.
- A pure population of agents running the competitive strategy tend to create coalitions of size equal to the number of requested skills.
- A pure population of agents running the conservative strategy tend to create small size coalitions.

– In a pure population of agents running the conservative strategy, there is a inversely proportional relationship between the size of coalitions and the degree of dynamism.

Lastly, the long term aim of this work is to analyse the clustering structure of agents within an RFP population - and investigate how this affects performance. To this end, Collaboration Graph representation seems to present a useful method to analyze properties of the Coalition Formation process in such long running scenarios.

8 Ongoing and Future Work

Ongoing and future work includes a deeper research on the conditions under which competitivity dominates conservative strategies.

Other planned work includes study of individual characteristics that make agents extraordinary, and extract the important patterns that are exploitable with certain strategies such as those shown in the current paper.

It will be also of significant value to study the stability of results from a formal game theoretic perspective, as well as the game theoretic properties of the strategies we study.

Finally, we are investigating the use of other large scale network analysis techniques to have a deeper understanding of our domain. Such techniques include *Clique Overlapping*, and *t-core distribution*.

References

1. T. Arnold and U. Schwalbe. Dynamic coalition formation and the core. *Journal of Economic Behavior & Organization*, 49:363–380, 2002.
2. R. Axtell. The emergence of firms in a population of agents: Local increasing returns, unstable nash equilibria, and power law size distributions. Working Paper 3, Center on Social and Economic Dynamics, Brookings Institution, 1999.
3. V. Batagelj and A. Mrvar. Pajek-program for large network analysis. http://vlado.fmf.uni-lj.si/pub/networks/pajek/.
4. P. Cailou, S. Aknine, and S. Pinson. A multi-agent method for forming and dynamic restructuring of pareto optimal coalitions. In *Proceedings of the 1st conference on Autonomous Agents and Multi-Agent Systems, AAMAS'02*, 2002. Bologna, Italy.
5. D. Cornforth, M. Kirley, and T. Bossomaier. Agent heterogeneity and coalition formation: Investigating market-based cooperative problem solving. In *Proceedings of the 3rd conference on Autonomous Agents and Multi-Agent Systems, AAMAS'04*, 2004. New York, USA.
6. A. Dragulescu and V. M. Yakovenko. Exponential and power-law probability distributions of wealth and income in the united kingdom and the united states. Computing in Economics and Finance 2002 125, Society for Computational Economics, July 2002.
7. D. D. Gatti and C. D. Guilmi. Financial fragility, industrial dynamics and business fluctuations in an agent based model. *paper presented ad the conference Wild@Ace 2003, Turin, Italy, October 3-4*, 2003.

8. D. D. Gatti, C. D. Guilmi, E. Gaffeo, G. Giuioni, M. Gallegati, and A. Palestrini. A new approach to business fluctuations: heterogeneous interacting agents, scaling laws and financial fragility, 2003.
9. M. Klusch and A. Gerber. Dynamic coalition formation among rational agents. *IEEE Intelligent Systems*, 17(3):42–47, 2002.
10. H. Konishi and D. Ray. Coalition formation as a dynamic process. *Journal of Economic Theory*, 110:1–41, 2003.
11. S. Kraus and O. S. ang G.Tasse. Coalition formation with uncertain heterogeneous information. In *Proceedings of the 2nd Conference on Autonomous Agents and Multi-Agent Systems, AAMAS'03*, 2003. Melbourne, Australia.
12. C. Merida-Campos and S. Willmott. Modelling coalition formation over time for iterative coalition games. In *Proceedings of the 3rd conference on Autonomous Agents and Multi-Agent Systems, AAMAS'04*, 2004. New York, USA.
13. V. L. Smith. An experimental study of competitive market behavior. *The Journal of Political Economy*, 70:111–137, 1962.

TAC-REM – The Real Estate Market Game: A Proposal for the Trading Agent Competition

Scott Buffett[1] and Maria Fasli[2]

[1] Institute for Information Technology - e-Business, National Research Council Canada, Fredericton, New Brunswick, Canada, E3B 9W4,
scott.buffett@nrc.gc.ca

[2] University of Essex, Department of Computer Science, Wivenhoe Park, Colchester CO4 3SQ, UK
mfasli@essex.ac.uk

Abstract. In this game, agents will face-off against each other in the ultra-competitive real estate market. Each competitor will act as a real estate agent, working on behalf of clients who need to move into new homes. These clients need to buy a new home as well as sell their current home. The game will test competitors' technology in two main research areas: preference elicitation and multi-issue negotiation. Each time an agent acquires a new client, it must query the client about its various preferences (e.g. price range, number of bedrooms, etc.) for their new home. Agents then search the listings of the other agents, seeking a possible match. Once found, the agent then engages in negotiations with the selling agent, haggling over various aspects of the deal. Once a house has been purchased, the client's old house needs to be sold. The objective of the game is to earn the most money. Selling agents earn commissions from sales. Buying agents do not earn commissions, but instead need to maximize the utility of their clients by obtaining a good deal. Satisfied clients are more likely to keep their agent to sell their old house.

1 Introduction

Research in automated negotiation and preference elicitation has been gaining momentum in recent years. While the traditional storefront e-commerce model with take-it-or-leave-it pricing has been the norm because of its relative ease, in purchase scenarios where several attributes related to the transaction (other than just price) need to be decided upon, a one-size-fits-all method of determining such values becomes too restrictive. Attributes must instead be tailored to the needs of the customer as well as to the availability of the seller. In such situations, negotiation becomes a valuable tool for searching the space of possible agreements for a mutually acceptable transaction. Given the speed with which transactions can be negotiated and executed through various electronic services today, research in intelligent agent technology has been focusing increasingly on automated negotiation [3,5,6]. Two of the main focuses of research in automated negotiation include the specification of protocols (i.e. the set of rules that the agents must follow during negotiation), and the construction of effective strategies (i.e. the rules agents use to determine how to operate within the protocols to achieve a successful or satisfactory outcome).

In order to construct an effective strategy, an agent must have a utility model over the set of possible outcomes of the negotiation. That is, the agent must have an idea as to the degree of preference over outcomes, so that it can work toward achieving an outcome that is highly preferred. If an agent is working on behalf of a human user (which is most often the case in electronic commerce), then it must have some idea of the user's preferences. Preference elicitation (also referred to as utility elicitation here) [2] is the field of research dedicated to studying the problem of effectively extracting these preferences or utilities from the user. The main issues here involve determining how to query the user, how to infer information from responses to queries, and how to determine when sufficient information has been elicited.

The Trading Agent Competition (TAC) [9] is an international competition where players submit autonomous trading agents that participate in a simulated market. In the original version of the game, referred to as "TAC Classic", agents participate in auctions to purchase components of potential vacation packages. Each agent works on behalf of several clients who have utilities for various travel packages. Agents attempt to satisfy their clients as much as possible while minimizing costs.

In 2003 the Supply Chain Management game (TAC-SCM) [1] was developed and added to the TAC event. In this game, agents procure computer components from suppliers, assemble the computers and sell them to customers. Supplier prices are set according to supply and demand, and computers are sold via auctions. The object of this game is to make the most money.

In this paper, we propose a new game to the series referred to as the Real Estate Market game (TAC-REM). In this game, agents work on behalf of clients to buy and sell houses. To determine which houses may be potential matches for their clients, or what transaction conditions may or may not be acceptable, agents must elicit their clients' preferences. Once a potential match is found, agents representing the potential buyer and seller enter into negotiations in order to find a mutually acceptable agreement. Commissions are earned on houses sold. The winner at the end of a game is the agent with the most money. While the first two TAC installments focused on auction and supply-chain technology, this new game will capitalize on agents' capabilities in preference elicitation and automated negotiation.

In addition to a new TAC game, the ideas given here have the potential to break new ground in the real estate business. Such technology can help pave the way for real estate companies and agents to explore the use of information technology and electronic commerce. Gibler and Nelson's [4] demonstration of the descriptive theories of how home buyers make choices shows that there is a gap between decision making in practice, and the more normative theory of choice as dictated by the axioms of utility theory [7]. This is largely because of the high complexity of quantifying a buyer's preference over the large number of attribute values that need to be considered when comparing houses. Instead, buyers often take an "elimination by aspects" approach [8]. Here, the most important attribute is chosen (in this case usually price), and an acceptable range of values is chosen. All alternatives falling outside of this range are then eliminated. Next the second-most important attribute is chosen, and so on. The decision-maker may iterate through this process several times, adjusting these ranges each time until a suitably-sized set of alternatives is found. Naturally, this decision-making strategy can provide sub-optimal results. By using our proposed system as simulation software, a real-estate agent can devise and test new intelligent methods for eliciting preference information from a client, and for using that information to make better decisions on which homes will make better matches.

In this preliminary paper, we present a general description of the game in section 2, followed by the game specification in section 3. This section gives details on such matters as daily activities, preference elicitation rules and procedures, utility functions and negotiation protocols, among others. Section 4 discusses a few issues that need to be resolved during the implementation and testing phases of the game production. In section 5 we close the paper with some final remarks.

2 Game Overview

The environment in which the agents compete is a typical real estate market. Each agent represents several clients, some of which are buyers and some of which are sellers. Buyers have preferences over the various attributes associated with potential houses. Such attributes include the number of bedrooms, number of bathrooms, square footage, type of neighbourhood, and whether or not it has a garage. Preferences for the combinations of various attribute values are modeled by a multi-attribute utility function. The agent representing the buyer must attempt to determine this utility function as accurately as possible by asking elicitation queries. There is little time to spend with clients, and each agent will likely have multiple clients, so the number of queries that can be made is limited. The agent must then find houses for sale that best meet the client's preferences.

Each selling client has a house for sale which is described using the values of the house attributes given above, with the addition that a price range is given as well. Each day, agents publish a listing of all of their clients' houses, accessible to all other agents.

A transaction is also made up of several attributes, including price, closing date, whether or not landscaping work will be done, whether or not maintenance will be performed on the heating/cooling system, and whether or not plumbing work will be done.

When an agent finds a house for sale that it believes will satisfy one of its buyers, it sets up a "showing". At this point, the client observes the attributes of the house and, based on its satisfaction with the house, computes and reveals its multi-attribute utility for the house, and indicates how likely it is that a potential transaction will meet or exceed its utility acceptability threshold. The seller of the house also has such a utility function and utility threshold for the various potential agreements. The two agents then enter into negotiations. Negotiations are bilateral where participants take turns exchanging one offer at a time, starting with the buyer. Each time an offer is received, the receiving agent can choose to either accept the offer, reject and counteroffer, or reject and quit.

If the agents are successful in achieving a deal, the transaction is made with the selling agent receiving a commission on the purchase price. The selling client then happily disappears from the market, and the agent representing the seller takes on a new buyer client. The client that bought the house now needs to sells its current home, and thus becomes a seller. It may or may not, however, retain the services of its current real estate agent. The client will choose to leave the agent with some probability depending on its utility of the negotiated transaction. The higher the utility, the higher the probability that it will stay with the current agent. If it chooses to leave the agent, it will choose another agent in the game randomly with uniform probability.

The objective of the game is to earn the most money. Money can only be earned through commission on selling houses. The exact percentage of the commission is computed based on how well the various aspects of the eventual deal satisfy the client. Thus the selling agents have an incentive to find acceptable deals not only with high prices, but also with other attribute values that will satisfy its client. Buyer agents receive no commission from transactions, but rather retain the buyer as a client based on the client's satisfaction. If the buyer is retained then it becomes a seller, making it a potential money-maker. So there is an incentive for buyer agents to find deals that satisfy their clients as well.

3 Game Specifications

3.1 Initial Setup

Six agents compete in a game. Initially each agent has 20 clients: 10 buyers and 10 sellers. Each buyer owns a home and wants to purchase a new one. Each seller simply has a home for sale (and presumably another one that they live in, but not for sale).

3.2 Summary of Daily Activities

1. *Obtaining new clients.* Each day begins with the registration of new clients. An agent receives new buyer clients in one of two ways: 1) by selling houses: each agent obtains a number of new buyer clients equal to the number of houses sold the previous day (i.e. each satisfied seller is replaced with a new buyer the next day), and 2) by obtaining other agents' unsatisfied buyer clients who were unable to buy a home after 14 days. An agent obtains new seller clients in one of three ways: 1) by purchasing a house for a buyer the previous day and convincing it to stay on as a seller client, 2) by obtaining other agents' buyer clients who left them after buying a home the previous day, and 3) by obtaining other agents' unsatisfied seller clients who were unable to sell their home after 14 days. See section 3.9 for more specific details.

2. *Update Listings.* Agents send the specifications of the houses that their clients have for sale to the central Multiple Listing Service. The central listing is updated once daily, early in the day. Entries can be submitted at any time, but those not submitted in time for the update will not appear until the next day. See section 3.3 for more specific details.
3. *Utility Elicitation.* Throughout the remainder of the day, agents can elicit utility information from their buyer clients to determine their preferences for different types of houses. Queries can ask about house and transaction preferences, and can offer two possible responses (e.g. "Would you prefer A or B'?"). To further ascertain utility, standard gambles may be included in the queries (e.g. "Would you prefer house type A or a gamble where you would receive house type B with 0.6 likelihood or house type C with 0.4 likelihood?"). See section 3.6 for more specific details.
4. *Showings.* A "showing" takes place when an agent allows a buyer client to assess its utility for a house that is for sale. At this point the client reveals its utility for the house (with average utilities used for the transaction attributes), and indicates the likelihood of being interested in buying the house, given its preferences for the likely transaction outcomes. An agent may choose to show its client a house belonging to one of its own selling clients, however the agent will incur penalties if it ultimately represents both clients in a transaction. See section 3.7 for more specific details.
5. *Negotiation.* If, after a showing, it is determined that there is an acceptable likelihood that an agreement on a transaction can be reached, then the agent may choose to enter in negotiations with the selling agent. Negotiations are bilateral, with the agent representing the buyer making the first offer. Each offer consists of a value for each transaction attribute. Negotiation continues until an agreement is reached or one of the agents quits. See section 3.8 for more specific details.
6. *Client Shuffle.* A client will find a new agent to work with if they are unsatisfied with their current one. Buyers and seller clients will stay with an agent for a maximum of two weeks (14 days). If the current agent has been unsuccessful in selling their current home or buying them a new one, they will leave the agent and enlist with a new one (randomly chosen uniformly). Also, a buyer client may leave its agent immediately after buying a new home, if it is somewhat unsatisfied with the purchase. The higher the utility of the purchase, the higher the probability that it will stay with its current agent. If it leaves, it will randomly choose a new agent. This shuffle phase takes place at the end of each day. No more of the above activities will take place once this phase begins. See section 3.9 for more specific details.

3.3 House Attributes, Listings

The house attributes are the attributes of a house that are non-negotiable, such as the size or the number of bedrooms. The house attributes and their domains are given in Table 1.

Table 1. Attributes and domains for houses

Attribute	Domain
House type	1-story, 2-story, split-level
Number of bedrooms	integer 1-5
Number of bathrooms	integer 1-4
Size	1000, 1500, 2000, 2500, 3000, 4000 sq ft
Neighbourhood type	urban, suburban, rural
Garage?	yes, no

The Multiple Listing Service (MLS) provides a public directory of all houses up for sale. In addition to the attribute values of each house, the listing provides information on the selling agent, the number of days on the market, and the price range. Example entries in the listing are given in Table 2.

Table 2. Example listing of available houses

Listing	Agent	Attributes	Price Range	Days on Market
0881	1	232421	[225000, 250000]	4
1021	6	343511	[250000, 275000]	2
1088	5	122230	[150000, 175000]	1

For example, the first entry indicates there is a house for sale with reference number 0881, agent number 1 is the selling agent, and the values for the attributes are as follows: the first attribute (type) has value 2 (2-story), the second attribute (# of bedrooms) has value 3 (3 bedrooms), the third attribute (# of bathrooms) has value 2 (2 bathrooms), the fourth attribute (size) has value 4 (2500 square feet), the fifth attribute (neighbourhood type) has value 2 (suburban) and the sixth attribute (garage?) has value 1 (yes). The price range is 225,000 - 250,000, and it has been on the market for 4 days. Note that all price ranges have size 25,000 with upper- and lower-limits being multiples of 25,000.

3.4 Transaction Attributes

The transaction attributes are those negotiable issues and conditions pertaining to the agreement of a sale, such as the price or whether landscaping work will

Table 3. Attributes and domains for transactions

Attribute	Domain
Price	integer 100000-500000
Preferred closing date	buyer's, seller's
Landscaping?	yes, no
Heating system maintenance?	yes, no
Plumbing maintenance?	yes, no

be done by the current owners as a condition of sale. The transaction attributes and their domains are given in Table 3.

3.5 Utility Functions

Each buyer client b has a utility function u^b for each house and transaction attribute. These utility functions are additively used to make up a buyer's multiattribute utility function that assigns a utility to each potential combination of house and transaction attribute values:

$$u^b(house, transaction) = \sum_{a \in H \cup T} w_a^b u_a^b(x_a) \tag{1}$$

where $H \cup T$ holds the house and transaction attributes, w_a^b is the weight of attribute a, u_a^b is the utility function for a, and x_a is the value for a given the house and transaction.

During the house-hunting process, a buyer's utility for a house is computed using the average utility for each transaction attribute, since the transaction attribute values are unknown and do not yet come into play. The particular values for these attributes are then used during the negotiation phase to determine the buyer's overall utility of a deal.

Each seller client s has a utility function u^s for each transaction attribute. These utility functions are additively used to make up a seller's multiattribute utility function that assigns a utility to each potential combination of transaction attribute values:

$$u^s(transaction) = \sum_{a \in T} w_a^s u_a^s(x_a) \tag{2}$$

where T holds the transaction attributes, w_a^s is the weight of attribute a, u_a^s is the utility function for a, and x_a is the value for a given the transaction.

For each type of client, the range of utilities for each attribute is [0,1], with the most favourable value yielding a utility of 1 and the least favourable yielding a utility of 0. All weights sum to 1, and thus the multiattribute utilities lie in the [0,1] range. When a represents price, u_a is not necessarily linear, but can rather be shaped to model the client's marginally decreasing (or increasing) utility.

Many attributes have a natural ordering in the preference of their values. For example, buyers always prefer smaller values for price while sellers always prefer larger values. As another example, buyers always prefer larger values for size than smaller values. It may be the case that buyers with limited funds prefer houses of smaller size, but this is likely because smaller houses are usually cheaper. All things being equal, a buyer will prefer a larger house to a smaller one. Values for other attributes such as neighborhood type do not have such a natural ordering. Some buyers may prefer urban living, while others prefer rural. Table 4 gives a summary of the orientation of each attribute.

Table 4. Orientation of attribute values

Attribute	Orientation
House type	none
Number of bedrooms	buyer prefers more
Number of bathrooms	buyer prefers more
Size	buyer prefers more
Neighbourhood type	none
Garage?	buyer prefers yes
Price	buyer prefers less, seller prefers more
Preferred closing date	each prefer their own
Landscaping?	buyer prefers yes, seller prefers no
Heating system maintenance?	buyer prefers yes, seller prefers no
Plumbing maintenance?	buyer prefers yes, seller prefers no

3.6 Preference Elicitation

In each elicitation query, the agent asks for the client's preference over two standard gambles g_1 and g_2. Each gamble g_i involves two potential deals d_i and d'_i, each made up of values for the house and transaction attributes, and a probability p_i. The uncertain outcome of g_i is d_i with probability p_i or d'_i with probability $1 - p_i$. So the query asks the following: "Would you prefer to take gamble g_1 where you would receive deal d_1 with probability p_1 or d'_i with probability $1 - p_i$, or would you prefer to take gamble g_2 where you would receive deal d_2 with probability p_2 or d'_2 with probability $1 - p_2$?" Values for all attributes are not required in the descriptions of d_1 and d_2; attributes with missing values will be ignored by the client. Agents can query about preferences over deals rather than gambles simply by making $p = 1$ and leaving d' blank.

Another type of query is an acceptability query. This is used when an agent wants to know whether a proposed agreement meets or exceeds the client's acceptability threshold. Such queries will typically be made during a negotiation. Here, the agent simply leaves g_2 blank. If the client indicates that the offer (or gamble) given in g_1 is preferred, then this means that g_1 is deemed acceptable to the client.

Table 5 shows how to formulate queries to elicit particular types of information, other than the general query for the client's preference over two standard gambles.

To determine a response to the query, the client uses its own secret utility function (described above) to compute the utility of each gamble:

$$u(g_i) = u(d_i)p_i + u(d'_i)(1 - p_i) \tag{3}$$

where $u(d_i)$ is computed using equation 1 for buyers and equation 2 for sellers. The client then indicates which of the two gambles has higher utility. In the case of a tie, g_1 is selected.

When an attribute value is missing, the client computes the overall utility of a deal using the average utility over all possible values for this attribute. When

Table 5. Formulation of query types (where d^+ and d^- are known to be the most and least preferred deals, respectively)

Query Type	g_1			g_2		
	d_1	d'_1	p_1	d_2	d'_2	p_2
Determine whether utility of deal d is $>$ or $<$ u	d^+	d^-	u	d	-	-
Determine which of deals d^A and d^B is preferred	d^A	-	1	d^B	-	1
Determine whether deal d is acceptable	d	-	-	-	-	-

querying about specific houses, the agent can enter the house reference number instead of entering all of the attribute values into the query.

3.7 Showings

When an agent believes that a house is a potential match for a buyer client, it may choose to "show" the house to the client. At this point, the client reveals its utility for the house (with average utilities used for the transaction attributes), and indicates the likelihood of being interested in buying the house. This likelihood is computed as the number of joint outcomes for transaction attribute values that meet its acceptability threshold (given the utilities of the house attribute values), divided by the total number of joint outcomes for transaction attribute values. Given this likelihood, the agent can decide whether to put in an offer (i.e. enter into negotiations with the seller agent), or to keep looking.

It may be the case from time to time that the best match for a buyer is a house that is being sold by a client represented by the same agent. An agent may represent both the buyer and the seller in a transaction, but will incur the following penalties:

1. 50% the commission earned from the sale
2. The buyer will leave the agent and choose to sell with another agent with 100% certainty

Thus such transactions are permissible but discouraged, and are likely only favourable as a last resort.

As showings take valuable time, each agent may show no more than 5 houses in a day, regardless of the number of clients it has.

3.8 Negotiating

A negotiation session begins with an agent representing a buyer sending an offer on a house to an agent representing the seller of that house. Each message in the negotiation consists of a house (given by the reference number), the intended buyer (given by the buyer number), and values for all transaction attributes. Each time an offer is received, the receiving agent may 1) accept, 2) reject and send a counteroffer or 3) reject and quit. An offer is valid for a fixed length of time.

Each time an offer is sent, the agent must first ensure that the client's utility for the offer meets or exceeds its acceptability threshold. This can be done using an acceptability query, or else can be concluded using past information, since a client's preferences do not change over time. If an agent ever accepts an offer that does not meet its client's acceptability threshold, or submits such an offer which is subsequently accepted, this agreement is still binding. The offending agent(s) in such a situation is (are) penalized an amount equal to the amount of money that could be offered to the client to make the deal acceptable. For example, consider an agent who agrees on a purchase of a house with price 200,000 on behalf of a buyer client with acceptability threshold u'. If the deal has utility for the buyer less than u', but a price of 190,000 (all else equal) would raise the utility of the purchase to u', then the buyer would be penalized $200,000 - 190,000 = 10,000$. The same holds true for sellers. In this case, sellers could effectively forgo some of their commission in order to help a client sell their house. Agents may have negative balances.

Since accepted offers are considered binding agreements, an agent may make only one offer at a time for any given client. That is, a seller agent may make an offer to only one potential buyer for a house that it is selling. No other offers may be made for that house until 1) the first offer expires, or 2) the agent that received the first offer rejects it or submits a counteroffer. The agent may entertain several offers from buyers for a house, thus effectively participating in several simultaneous negotiations for the house, but may only have one outstanding offer from itself at any given time. The same goes for buyer agents who represent a client that is interested in more than one house. At any time, the agent may submit at most one offer on behalf of the buyer. Naturally, an agent can represent several buyers that are interested in the same house.

3.9 Obtaining/Losing Clients

Each time an agent successfully sells a house for a selling client, that client disappears from the game, and the agent receives a newly-produced buyer client the next day. When an agent successfully purchases a house for a buyer client, that client turns into a seller client the next day. However, if the client was not particularly satisfied, it may choose to leave the agent and find a new one. Let u' be the minimum utility deemed acceptable by the client and $u(d)$ be the utility achieved by the sale given by d. If $u(d) = u'$ (the worst case), the agent has a 50% chance of losing the client. If $u(d) = 1$ (the best case), the agent has a 0% chance of losing the client. The probability for all other $u(d)$ is computed by equation 4. The function from equation 4 is depicted in Figure 1. Note that $u(d)$ will never really be less than u'. Whenever an agent makes a deal where this is the case, the agent is penalized the amount that effectively brings $u(d)$ up to u'.

$$p(lose) = \frac{(\frac{u(d)-u'}{1-u'} - 1)^2}{2} \tag{4}$$

Using this type of function (as opposed to a more linear function) ensures that the probability of losing a client is relatively low in most cases, but increases sharply if the agent exerts minimal effort and barely satisfies the client. This

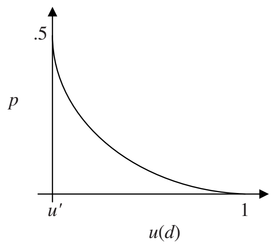


Fig. 1. Probability of a buyer client leaving its agent

discourages the agent to simply agree to the first acceptable offer so it will have time to move on to other tasks.

Both buyer and seller clients will stay with the same agent for a maximum of 14 days. At the end of the 14th day, if the agent has not either purchased (in the case of the buyer) or sold (in the case of the seller) a house for the client, the client will leave the agent. A client that leaves an agent will enlist with a randomly chosen agent at the beginning of the next day. This client retains its original preferences, and thus the client's former agent may retain information learned about these preferences for possible later use. After an additional 14 days, if the client has still not completed a transaction, it is deemed to be not satisfiable and is removed from the system. It is then replaced with a new client of the same type and is assigned to an agent (possibly the current one) randomly.

When a buyer converts to a seller and stays with the same agent, this day counter is reset, with the next day (the first day that the agent can sell for them) being day 1.

3.10 Commission

The selling agent receives a commission from each sale. This is the only way to earn money in the game. The seller earns a base commission of 5% of the sale price for making the sale, plus a bonus of up to 5% more based on client satisfaction. The total commission c earned for deal d with sale price s is computed by

$$c = (0.05 + 0.05(\frac{u(d) - u'}{1 - u'}))s \tag{5}$$

where u' is the minimum utility deemed acceptable by the client and $u(d)$ is the client's utility for d.

3.11 Duration

The game lasts 365 days. At the end the winner is chosen to be the agent that made the most money.

4 Future Work

At this point in the game development, there are several issues that remain unresolved. For the most part, such resolutions can reasonably be made only after implementation and experimentation. We mention some of these issues here.

- *Creating clients.* New clients are created randomly, but must be done so as to ensure fairness among the competitors. Clearly certain agents will have an advantage if they always receiver buyer clients that are easy to please and seller clients with premium houses.
- *Reputation.* Under the current game specification, when a client leaves an agent it selects a new agent randomly with uniform probability. To make the game more realistic, we hope to introduce the concept of reputation. Agents' reputations will be based on how well they are able to satisfy their clients. Those agents with high reputation would then be more likely to land new clients.
- *Running time.* The game should not run longer than 60 to 70 minutes, which is comparable to the TAC Classic and TAC-SCM games. Otherwise it will be impractical to run in a competition setting. The length of a day would then be 1/365 of the game time.
- *Number of queries per day.* In order to be more realistic, and encourage the use of intelligent methods for determining optimal queries and inferring information from query responses, the number of times an agent is allowed to query a client should be limited. It may be the case that number of possible queries is limited by the short time that is available for doing so each day. Otherwise a hard cap of, say two queries per day per client, may have to be imposed.
- *Stages for each activity.* It may be best that each of the major activities that take place during the course of a day are divided into stages. For example, the day could begin with the listing stage, followed by the elicitation stage, followed by the showing stage, followed by the negotiation stage. Only the specified activity can be done during each stage. This will likely make things easier from an implementation perspective, while imposing a reasonable workflow on the agents' daily activities.
- *Offer expiry.* Offers cannot be valid indefinitely. An agent might not get an immediate response from the receiver of the offer, and cannot be expected to wait very long before looking elsewhere. This issue will be best resolved during the testing phase of the project.

5 Final Remarks

This paper gives a proposal for a third edition to the TAC series, called the Real Estate Market game (TAC-REM). Agents work on behalf of clients to buy and sell houses. To determine which houses may be potential matches for their clients, or what transaction conditions may or not be acceptable, agents must elicit their clients' preferences. Once a potential match is found, agents representing the potential buyer and seller enter into negotiations in order to find a mutually acceptable agreement. Commissions are earned on houses sold. The objective of the game is to make the most money over a 365-day period.

This paper serves as an invitation to the TAC community to consider the game presented in this paper, and provide us with feedback and suggestions on how to improve it. The game should provide an excellent forum for researchers in the areas of automated negotiation and preference elicitation to test their techniques and solutions. It may also pave the way for real-world real estate companies to explore the use of information technology and electronic commerce to conduct business. Above all, the game promises to be interesting and engaging, and thus has high potential to promote research in these areas by attracting and encouraging new researchers and students to participate.

References

1. Raghu Arunachalam and Norman Sadeh. The 2003 supply chain management trading agent competition. In *Proc. of the 6th International Conference on Electronic Commerce (ICEC2004)*, pages 113–120, Delft, The Netherlands, 2004.
2. U. Chajewska, D. Koller, and R. Parr. Making rational decisions using adaptive utility elicitation. In *AAAI-00*, pages 363–369, Austin, Texas, USA, 2000.
3. S.S. Fatima, M. Wooldridge, and N. R. Jennings. An agenda-based framework for multi-issue negotiation. *Artificial Intelligence*, 152(1):1–45, 2004.
4. K. M. Gibler and S. L. Nelson. Consumer behavior applications to real estate education. *Journal of Real Estate Practice and Education*, 6(1):63–84, 2003.
5. N. R. Jennings, P. Faratin, A. Lomuscio, S. Parsons, C. Sierra, and M. Wooldridge. Automated negotiation: prospects, methods and challenges. *Int. J. of Group Decision and Negotiation*, 10(2):199–215, 2001.
6. N. R. Jennings, S. Parsons, C. Sierra, and P. Faratin. Automated negotiation. In *5th International Conference on the Practical Application of Intelligent Agents and Multiagent Systems (PAAM-2000)*, pages 23–30, Manchester, UK, 2000.
7. R. L. Keeney and H. Raiffa. *Decisions with Multiple Objectives: Preferences and Value Tradeoffs*. John Wiley and Sons, Inc., 1976.
8. A. Tversky. Elimination by aspects: a theory of choice. *Psychological Review*, 79:281–299, 1972.
9. MP Wellman, PR Wurman, K O'Malley, R Bangera, S d Lin, D Reeves, and WE Walsh. Designing the market game for a trading agent competition. *IEEE Internet Computing*, 5:43–51, 2001.

Evolutionary Stability of Behavioural Types in the Continuous Double Auction

Perukrishnen Vytelingum, Dave Cliff, and Nicholas R. Jennings

School of Electronics and Computer Science,
University of Southampton, Southampton, SO17 1BJ, UK
{pv03r,dc,nrj}@ecs.soton.ac.uk

Abstract. In this paper, we investigate the effectiveness of different types of bidding behaviour for trading agents in the Continuous Double Auction (CDA). Specifically, we consider behavioural types that are *neutral* (expected profit maximising), *passive* (targeting a higher profit than neutral) and *aggressive* (trading off profit for a better chance of transacting). For these types, we employ an evolutionary game-theoretic analysis to determine the population dynamics of agents that use them in different types of environments, including dynamic ones with market shocks. From this analysis, we find that given a symmetric demand and supply, agents are most likely to adopt neutral behaviour in static environments, while there tends to be more passive than neutral agents in dynamic ones. Furthermore, when we have asymmetric demand and supply, agents invariably adopt passive behaviour in both static and dynamic environments, though the gain in so doing is considerably smaller than in the symmetric case.

1 Introduction

The last decade has seen a significant change in the nature of electronic commerce with the emergence of trading agents [6], software that is capable of autonomous and flexible action to achieve its objectives and that is endowed with sophisticated strategies for maximising profit in different types of market mechanisms. Now, one of the most important such mechanisms is the Continuous Double Auction (CDA) [4], a symmetric auction with multiple buyers and sellers. CDAs are so important because they are the principal financial institution for trading securities and financial instruments (e.g. the NYSE and the NASDAQ both run variants of the CDA institution). However, developing agents that can participate in the CDA is difficult because it is not amenable to a game-theoretic analysis and there is no known optimal strategy [4]. Therefore, a number of heuristic strategies have been proposed [12,11,14], each of which has a particular behaviour in the market. Given this, in this paper, we are not concerned with developing yet another strategy, but rather we are interested in how a particular characteristic of such strategies impacts upon their behaviour and their effectiveness. The characteristic in question is the *aggressiveness* of the bidding behaviour, here defined as how eager an agent is to transact. We focus on aggressiveness in particular because we believe it is a key determinant of success in the market. In particular, we consider three behavioural types[1]:

1. The *neutral* agent always submits the quote (a bid or an ask) that maximises its expected profit. This is the most common type of behaviour and is one that is often hardwired into various strategies.
2. The *passive* buyer (seller) submits a lower (higher) quote than its neutral counterpart in order to try and obtain a more profitable transaction. Thus if it does transact, it makes more money because it pays less (if it is a buyer) or receives more (if it is a seller).
3. The *aggressive* buyer (seller) submits a higher (lower) quote than its neutral counterpart in order to try and improve its chances of transacting as it offers more (if it is a buyer) or asks for less (if it is a seller) with the aim of making sure that it can trade.

At this time, comparatively few researchers have considered the issue of varying aggressiveness in a market context. Abreu *et al.* [1] describe the evolutionary stability of behaviours in a reputational model of bargaining, and how players can be more profitable when adopting passive or aggressive behaviour. Thus, while they show that this attitude can have a significant effect on the outcomes experienced, they do not consider the CDA. Walsh *et al.* do consider the CDA and they use an evolutionary game-theoretic (EGT) analysis to examine the interaction of a number of common strategies [15,16]. Thus, their analysis provides an insight on the population proportion that will adopt each strategy. But, while they restrict their analysis to a particular set of well-known strategies, we are more interested in whether a particular strategy can perform better if it is passive, neutral or aggressive. Phelps *et al.* [9] also use an EGT analysis, but they compare two double auction mechanisms, the call and continuous, given that similar strategies are available for both mechanisms. Thus, they are interested in the performance of double auction mechanisms given particular strategies (including one that is evolved using a GA).

Against this background, we believe this study of behavioural types is important because it provides an insight into how this fundamental aspect of bidding behaviour affects the system's performance. Thus, these results apply to any CDA strategy that is capable of adjusting its behaviour along this dimension (e.g. GD [5], ZIP [3] or RB [14]). Such insights are important because if an agent can be more profitable by deviating to another behaviour, then it will do so. However, with every agent in the market doing this, the population distribution of types can change significantly. Now, an effective trading agent can use knowledge of such dynamics to decide on what behavioural type to adopt given the

[1] The nomenclature for the different behaviours varies over the literature, with some like Abreu *et al.* [1] considering passive and aggressive behaviours and others like Byde [2] considering the behaviour of different risk attitudes. However, they all refer to some similar behaviour. For example, a risk-averse agent adopts aggressive behaviour to improve its chance of winning, while a risk-seeking agent adopts passive behaviour to target higher profits. The risk neutral agent adopts a neutral behaviour.

particular population distribution of types. Furthermore, because behavioural aggressiveness is usually an endogeneous aspect of a strategy, such an analysis can assist the strategy designer when assessing the effectiveness of various strategies. To perform our analysis, we adopt a similar EGT approach to that of Walsh *et al* and we choose EGT because it allows us to study the dynamics of the CDA when agents are allowed to evolve in terms of the behaviour they adopt. To do this, however, we need to describe the model in a form that abstracts the complex bidding that the CDA mechanism entails, into a simple normal-form game. By so doing, the CDA then becomes amenable to such an analysis. In particular, we develop a set of strategies that vary only in terms of their different behavioural types.

In Section 2, we formally define the CDA mechanism we use in this work and then in Section 3 we describe how we model the behavioural type of a strategy. In Section 4, we detail the model we use for our EGT analysis and in Section 5 we provide our actual analysis. Section 6 concludes.

2 The Continuous Double Auction

The CDA mechanism allows agents to submit quotes at any time during the auction, and a transaction can occur whenever a buyer's bid and a seller's ask can be matched. We conform to previous studies on the CDA in terms of our experimental setup [10,12]. In particular, a set of limit prices is endowed to buyers and sellers and these determine the market demand and supply. Thus, a desired demand and supply can be induced with the appropriate endowment to get the desired environment. Furthermore, the CDA consists of a number of fixed-duration trading periods (or trading days) at the beginning of which an agent is given an endowment to buy or sell. The CDA protocol includes the *NYSE spread-improvement rule* requiring that any bid (ask) submitted must be higher (lower) than the outstanding bid[2] (ask). Because of the non-deterministic nature of the CDA, payoffs to trading agents are averaged over a sufficient number of simulations to obtain statistically significant results, as computed by the Wilcoxon rank sum test (see Section 5).

3 Modelling the Behavioural Types

Software trading agents are usually designed to employ neutral bidding strategies. However, as discussed in Section 1, we are interested in the effect on the market if trading agents are also designed to be non-neutral and instead have passive or aggressive bidding behaviour. To this end, we model such behaviours by modifying the widely used GD strategy [5]. We choose this strategy out of all those available [12,3,14] because it is one of the most efficient [12] and it can readily be extended to incorporate different bidding behaviours. However, we

[2] The outstanding bid, o_{bid}, is the current highest bid and the outstanding ask, o_{ask}, is the lowest ask in the market.

could equally well have chosen any CDA strategy that can be suitably adjusted to model passive or aggressive behaviour, and our results generalise to these cases (not shown due to space restrictions).

Now, in the original GD strategy, while all GD agents have the same belief about the market, they form a neutral expected utility-maximising quote based on their own private preferences. In this section, therefore, we detail the GD strategy and how we modify it to be either passive or aggressive (in addition to its standard neutral perspective).

In more detail, the GD agent has a belief that its quote to buy or sell will be accepted in the market. The agent then submits the price that maximises its expected utility. The utility of a quote is the profit associated with such a quote given the agent's private preferences (limit price, ℓ). The GD agent forms its belief on the basis of observed market information, H_M, that includes the frequencies of accepted quotes (transactions) and rejected quotes. Furthermore, the belief encompasses some form of recency so that only the most recent quotes during the past L transactions are considered. We now formally state the buyer's belief, $\widehat{q}(b)$, associated with a bid b, and seller's belief, $\widehat{p}(a)$, associated with an ask a:

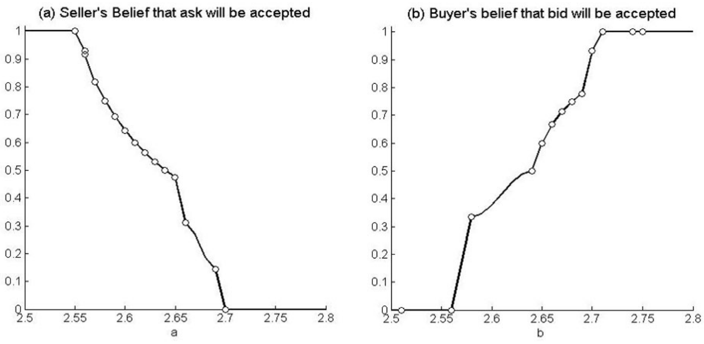


Fig. 1. (a) Seller's belief, $\widehat{p}(a)$, that an ask a will be accepted in the market. (b) Buyer's belief, $\widehat{q}(b)$, that a bid b will be accepted in the market.

Definition 1. ***Bid Frequencies:*** *D is the set of all permissible quotes in the market. $\forall d \in D$, $B(d)$ is the total number of bid quotes made at price* d*, $TB(d)$ is the frequency of accepted bids at* d*, and $RB(d)$ the frequency of rejected bids at* d*.*

Definition 2. ***Ask Frequencies:*** *D is the set of all permissible quotes in the market. $\forall d \in D$, $A(d)$ is the total number of ask quotes made at price* d*, $TA(d)$ is the frequency of accepted asks at* d*, and $RA(d)$ the frequency of rejected asks at* d*.*

$$\widehat{q}(b) = \frac{\sum_{d \le b} TB(d) + \sum_{d \le a} A(d)}{\sum_{d \le a} TB(d) + \sum_{d \le a} A(d) + \sum_{d \ge a} RB(d)} \tag{1}$$

$$\widehat{p}(a) = \frac{\sum_{d \geq a} TA(d) + \sum_{d \geq a} B(d)}{\sum_{d \geq a} TA(d) + \sum_{d \geq a} B(d) + \sum_{d \leq a} RA(d)} \tag{2}$$

We use a cubic spline interpolation to calculate the belief at prices that are not registered in H_M (see [5] for further details). The seller's belief function is then modified to satisfy the *NYSE spread-improvement rule* (see Section 2). Thus, for any ask that is higher than the current outstanding ask, the belief function is set to 0. Similarly for the buyer, the belief that any bid below the outstanding bid is accepted is 0. An example of a buyer's and a seller's belief function is shown in Figure 1.

Given its belief function and neutral behaviour, the GD agent forms a quote that maximises its expected utility. In particular, the buyer's utility, $U_b(b)$, for a bid b is given in Equation 3 and the seller's utility, $U_s(a)$, for an ask a in Equation 4. In both cases, the expected utility is the product of the belief function and the utility function, and the bid b^* or ask a^* to submit is computed as the price that maximises the expected utility, as shown in Equation 5.

$$U_b(b) = \begin{cases} 0 & \text{if } b \leq \ell \\ (\ell - b) & \text{otherwise} \end{cases} \tag{3}$$

$$U_s(a) = \begin{cases} 0 & \text{if } a \leq \ell \\ (a - \ell) & \text{otherwise} \end{cases} \tag{4}$$

$$\begin{aligned} b^* &= \arg\max_{b \in (o_{ask}, o_{bid})} \left[U_b(b)\widehat{q}(b)\right] \\ a^* &= \arg\max_{a \in (o_{ask}, o_{bid})} \left[U_s(a)\widehat{p}(a)\right]. \end{aligned} \tag{5}$$

To modify the GD strategy so that it embodies the other behavioural types, we alter the agent's utility as a function of its limit price, ℓ, and some scalar parameter, θ, that represents its aggressiveness in the market. The new aggression-sensitive utility functions, $\widetilde{U}_b(b)$ and $\widetilde{U}_s(a)$ for buyers and sellers respectively, are given in Equations 6 and 7. Here, we vary θ from negative to positive to describe passive to aggressive behaviour, with $\theta = 0$ describing neutral behaviour. By way of an illustration, Figure 2 shows the different utility functions for passive ($\theta = -1$), neutral ($\theta = 0$) and aggressive behaviour ($\theta = 1$) for the buyer and the seller.

$$\widetilde{U}_b(b) = \begin{cases} 0 & \text{if } b \leq \ell \\ (\ell - b)e^{\frac{\theta b}{\ell}} & \text{otherwise} \end{cases} \tag{6}$$

$$\widetilde{U}_s(a) = \begin{cases} 0 & \text{if } a \leq \ell \\ 1 & \text{if } a > a_{max} \\ (a - \ell)e^{\frac{\theta(a_{max} - a)}{a_{max} - \ell}} & \text{otherwise} \end{cases} \tag{7}$$

Before we proceed to our EGT study of behaviours, we evaluate our new space of GD strategies to ensure that they remain reasonably efficient compared

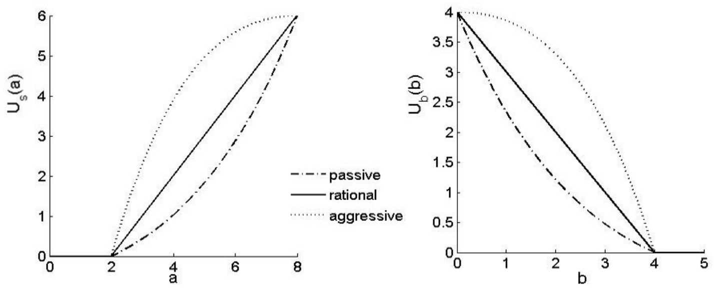


Fig. 2. (a) Seller's utility $U_s(a)$ for an ask a and $\ell = 2.0$. (b) Buyer's utility $U_s(b)$ for a bid b and $\ell = 4.0$. The utility is given as a function of the agent's limit price, ℓ, and the aggressiveness in bidding behaviour. $\theta = -1$ describes passive behaviour, $\theta = 0$ neutral behaviour, and $\theta = 1$ aggressive behaviour.

to their neutral counterpart. We do so in a homogeneous environment with a typical symmetric demand and supply. Specifically, we examine the market efficiency[3] of the CDA with a 10-buyer and 10-seller homogeneous population with similar behaviour. To this end, Figure 3 gives the market efficiency over the aggressiveness parameter θ and different numbers of allocations to buy or sell. The latter gives an insight on how efficiency changes as the number of expected transactions in the market increases with larger allocations given to the traders. As can be seen, as the market size increases with the number of tradeable units, the market efficiency increases for all behaviours. However, *all* agents adopting neutral behaviour does indeed maximise market efficiency in all cases. Passive behaviour is slightly less profitable while aggressive behaviour performs worst. Thus, Gjerstad and Dickhaut's decision to make GD neutral is validated in such a homogeneous environment (and indeed this is the result that they report in [5]). However, we cannot assume that all agents will remain neutral, as if it profits an agent to deviate to another behaviour, it will do so.

Given this, the question that naturally follows is whether an agent can indeed improve its performance by adopting a different behavioural type. Now, to answer that question, we use an evolutionary game-theoretic analysis, which we describe in the next section.

4 An Evolutionary Game-Theoretic Model

Evolutionary game-theory has traditionally been used to analyse simple games (such as the Prisoner's Dilemma) in terms of the dynamics of the population of learning agents playing different strategies [17]. Here, however, we are interested in a much more complex game and so a slightly different model is needed. In particular, as explained in Section 1, we use Walsh *et al.*'s model which involves

[3] The market efficiency is the ratio of actual total profit of *all* agents to the maximum profit that could be extracted in the market.

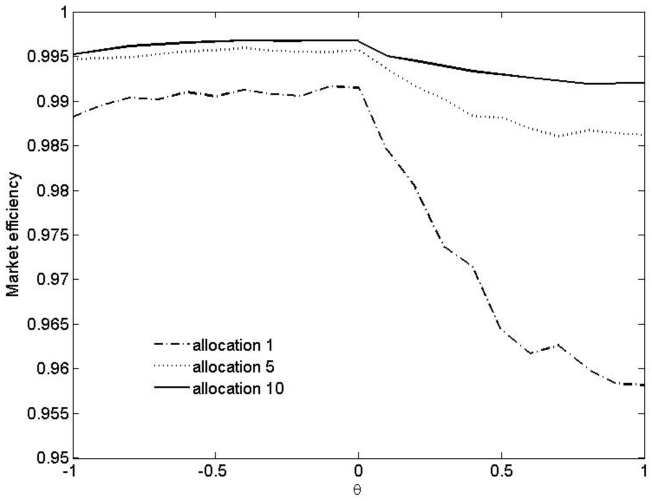


Fig. 3. Market efficiency given a homogeneous population of agents with similar behavioural type. We consider 3 different set of experiments with agents allocated 1, 5 and 10 limit prices.

considering the action of an agent as playing its bidding strategy during a game (lasting several trading days) and the payoff as being the total profit at the end of the game. Furthermore, they describe these payoffs as being *heuristic* because they are the output of a mapping of the strategies through a complex, non-deterministic interaction of the trading agents.

Given this, we first describe how we compute the heuristic payoff table that details the expected payoff to each agent (as a function of the S strategies that agents are allowed to play, and the combination of the A agents playing those strategies). We then describe how we use this table to compute the mixed Nash equilibrium of the game and the well-documented *replicator dynamics* model [17] (which is a standard way of representing the population distribution changes).

4.1 Computing the Heuristic Payoff Table

With the heuristic payoff table, we are interested in the expected payoff of a player playing a strategy, j, given the strategies adopted by the other $(A-1)$ players. Now, because of the non-deterministic and complex nature of the CDA game, some simplifications are required:

1. The payoff of a strategy is the average payoff of an agent playing that strategy in the CDA game, given the different strategies all the A agents are playing in that game.
2. All agents have the same set of strategies to play, and have the same payoff when playing the same strategy. Thus, as described in [16], we can restrict our analysis to symmetric games [17], and significantly reduce the complexity of the problem. Rather than having a table of size S^A, we reduce it to $\binom{A+S-1}{A}$ entries.

Given these, we build our heuristic payoff table[4] by considering the exhaustive set of strategies the A agents can play, and the number of agents playing each strategy (rather than considering which strategy each of the A agents is playing). Now, because payoff in the CDA game is non-deterministic, we require a significant number of independent simulations for each table entry to ensure that these are statistically significant. Thus, for each entry, we run a number of CDA games (typically 1000) with A agents, each assigned a strategy and a type (buyer or seller) to play, ensuring there is an equal number of buyers and sellers, with a probability of 0.5 that there will be an additional buyer or seller if A is odd.

Given our payoff table, we can now proceed with an EGT analysis as we would with a normal-form game.

4.2 Computing the Equilibrium

Here, we describe how to compute the mixed Nash equilibrium of the CDA game. An agent i chooses the strategy it plays according to its *mixed-strategy*, $\hat{p}_i = (\hat{p}_{i,1}, ..., \hat{p}_{i,S})$ and $\sum_{j=1}^{S} \hat{p}_{i,j} = 1$, where $\hat{p}_{i,j}$ represents the probability that agent i plays strategy j. At the equilibrium, $\hat{p}_i^*$, an agent i cannot receive a higher payoff by unilaterally deviating to another mixed-strategy, assuming that the other agents do not change their strategies [17]. Now, because we assume a symmetric game, all agents have the same mixed-strategy and the same mixed Nash equilibrium. Furthermore, because we are considering a very large population, we can validate that p is equal to the mixed-strategy. Given this, we denote both the population distribution and the mixed-strategy as $p = (p_1, ..., p_S)$, and the mixed Nash equilibrium as p^* hereafter.

In our EGT analysis, we denote the expected payoff of an agent playing a strategy j, given the mixed-strategy p, as $u(e^j, p)$. To compute $u(e^j, p)$, we consider the results from a large number of CDA games with an agent playing strategy j and $(A-1)$ agents selected from the population, with a mixed-strategy p. For each game and every strategy, we average[5] the individual payoffs (obtained from our heuristic-payoff table) of agents using strategy j. The mixed Nash equilibrium is then formulated as the argument to the minimisation problem given in Equations 8 and 9. Specifically, p^* is a mixed Nash equilibrium if and only if it is a global minimum of $v(p)$ [16,8], and we can validate that p is a global minimum if $v(p) = 0$.

$$v(p) = \sum_{j=1}^{S} \left(\max\left[u(e^j, p) - u(p, p), 0\right]\right)^2 \tag{8}$$

[4] A table entry for a 20-player game with 3 strategies would be $(|S_1|, |S_2|, |S_3|, U_1, U_2, U_3)$ where S_j is the set of agents playing strategy j, $|S_j|$ is the number of agents playing strategy j, and U_j is the average payoff of an agent playing strategy j. Note $\sum_{j=1}^{3} |S_j| = 20$ and we have 231 entries.

[5] In effect, here, we are not running the CDA game for every simulation, but only selecting the appropriate payoff from our heuristic payoff table each time.

where $u(p, p) = \sum_{j=1}^{S} u(e^j, p)p_j$ is the average payoff of an agent in a population with distribution p.

$$p^* = \arg\min_{\overline{p} \in \Delta} [v(p)] \tag{9}$$

Solving such a non-linear minimisation problem is non-trivial and computationally demanding. Thus, we use a method provided by the Matlab optimization toolbox based on the Nelder-Mead method to find the zero-points of the function v. Because the algorithm used is a non-linear *local* minimiser, we restart the algorithm repeatedly at random points within the unit-simplex until no new equilibria are found for 20 runs.

Now, while the mixed Nash equilibrium gives a theoretical and static perspective of our simplified CDA game, the dynamics of the game and how the equilibria are reached often provide more insight. Given this, we turn to the replicator dynamics which have been shown to be a good model for a common kind of agent learning (namely Reinforcement Learning) [13] and describes how agents learn to reach the equilibrium.

4.3 Computing the Replicator Dynamics

The replicator dynamics, $\dot{p} = (\dot{p}_1, ..., \dot{p}_S)$, describe how the population distribution p (where $p = (p_1, p_2, ..., p_S), p \in \Delta$ is an element of a unit-simplex, Δ, and $\sum_{j=1}^{S} p_j = 1$) changes. This approach assumes that an agent deviates to another strategy that appears to be receiving a higher payoff. Specifically, $\dot{p}$ is a vector given as follows:

$$\dot{p}_j = \left[u(e^j, p) - u(p, p)\right] p_j \tag{10}$$

The replicator dynamics show us the strategy trajectories and how they converge to an equilibrium, though they do not necessarily settle at a fixed point (see [17]). In this context, an equilibrium to which trajectories converge, and settle, is known as an *attractor*, while a *saddle point* is an unstable equilibrium at which trajectories do not settle. The region within which all trajectories converge to a particular equilibrium is known as the *basin of attraction* of that equilibrium. The basin is a very useful measure of the adoption of the attractor equilibrium and how likely the population is to converge to that equilibrium.

Here, we compute $\dot{p}$ by starting at different population distributions p inside the unit-simplex and following the trajectory given by Equation 10.

5 An Empirical Analysis of the Behavioural Types

In these experiments, we compute the heuristic-payoff table for a 20-agent CDA game with 3 strategies and lasting 10 trading days. At the beginning of each day, buyers and sellers are each endowed with a single limit price drawn from uniform distributions U_b and U_s respectively. For the purposes of this paper, we consider different uniform distributions to model representative (symmetric and asymmetric) small markets (similar to those considered in previous studies on the CDA [5,3,14]):

- Market 1: $U_b = \mathcal{U}(1.5, 4.5)$ and $U_s = \mathcal{U}(1.5, 4.5)$. This is a symmetric market that has an expected equilibrium at 3.0.
- Market 2: $U_b = \mathcal{U}(2.5, 5.5)$ and $U_s = \mathcal{U}(2.5, 5.5)$. This is a symmetric market that has an expected equilibrium at 4.0.
- Market 3: $U_b = \mathcal{U}(1.5, 4.5)$ and $U_s = \mathcal{U}(2.8, 3.2)$. This is an asymmetric market in which the slope of the supply curve is greatly reduced (compared to market 1). The equilibrium is expected at 3.0.

We model a market shock by changing the demand and supply on trading day 6 (and the new demand and supply then remain the same for the ensuing trading days). In particular, we consider two market shocks, *MS12* where demand and supply changes from Market 1 to 2 and *MS13* where it goes from Market 1 to 3. The former is a market shock that simply results in an increase in equilibrium price, but produces no change in the symmetry of market. The latter is a more complex shock as the structure of the demand and supply changes, though the equilibrium price does not.

We split our analysis into static and dynamic environments to compare our interpretations of their dynamics. For the former, we consider one symmetric market (1) (the results are similar for Market 2) and the asymmetric market (3). For the latter, we consider both types of market shocks. We validate our result by running a Wilcoxon rank sum test [7] on randomly selected entries covering 10% of our heuristic payoff table in order to ensure statistical significance of our data. In some experiments where the change across the heuristic payoff table is particularly small (e.g. in Market 3), 2000 simulations were required for significance.

Now, because we are considering only 3 strategies (each corresponding to a behavioural type), the population distribution space is a unit-simplex in a three-dimensional space. Thus, we can visualise the replicator dynamics $\dot{p}$ and the equilibria p^* by projecting the simplex onto a two-dimensional space [17]. The contour shading in our simplex is proportional to the magnitude of $\dot{p}$.

5.1 Static Environments

We first analyse behaviours in the CDA when there are no market shocks. In Market 1, we have 2 attractors at A and C and a saddle point at B (see Figure 4). As can be seen, the basin of attraction of C is considerably larger than that of A. This means the majority of the trajectories converge to neutral behaviour, with a non-significant proportion settling at passive. We can see that neutral behaviour is evolutionarily stable in symmetric markets and so most agents are likely to gravitate towards it. Thus, as a designer in such situations, the best thing is to do is make your agent neutral.

Now, when we consider the dynamics in asymmetric Market 3, we find a single attractor at A (see Figure 5). Thus, all trajectories converge towards passive behaviour. However, we note that the magnitude of the dynamics is considerably smaller compared to that in Market 1. This reflects the comparatively small gain that the agent achieves in Market 3 as it slowly adopts the evolutionary

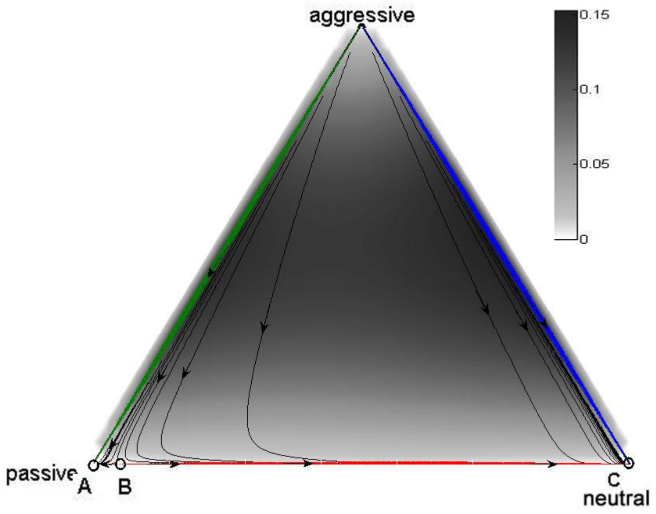


Fig. 4. The replicator dynamics of a CDA game in Market 1. Here, we have 3 equilibria with attractors at A=$(1, 0, 0)$ and C=$(0, 1, 0)$ and a saddle point at B=$(0.95, 0.05, 0)$.

stable passive behaviour. Here, we can interpret the small gain as follows. At a particular population distribution, the agent might intend to deviate in opposing directions from a buyer's and a seller's perspective, particularly because of the asymmetry in the market. Because an agent can be either a buyer or a seller with the same behavioural type, the deviation is a composite of the buyer's and seller's intentions, which explains the small gain as the dynamics converge. Having said this, however, it is still more profitable for an agent to be passive in asymmetric markets, and it should be designed as such.

5.2 Dynamic Environments

The neutral GD strategy has been shown to react well to simple symmetric market shocks [5] in a homogeneous environment. Here, however, we are interested to see whether agents will still adopt neutral behaviour in the presence of market shocks, or whether there are behaviours that are more profitable in such dynamic environments.

With market shock *MS12*, we have 2 attractors at A and C, and a saddle point at B (see Figure 6), with a relatively larger basin of attraction for equilibrium A. Here, there are more agents that are likely to be passive than neutral, though they might all adopt neutral behaviour if there are insufficient passive agents in the market. Thus, it can be more profitable not to be neutral all the time when there are simple market shocks. The strategy designer can use the knowledge of how the dynamics reach the equilibrium to decide what strategy the agent should adapt given the population distribution of behaviours.

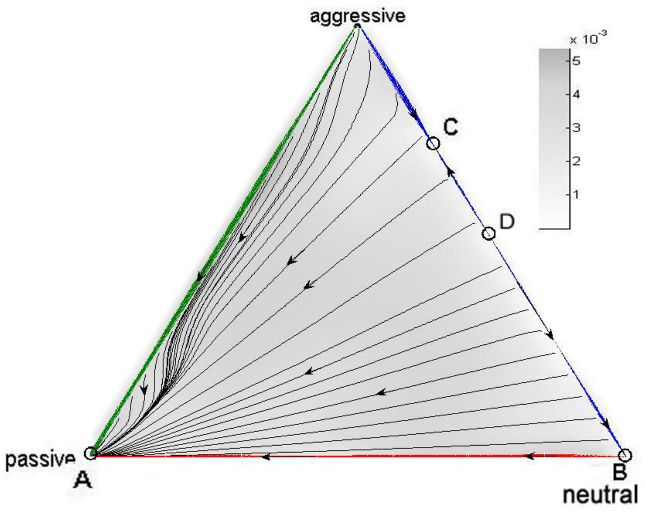


Fig. 5. The replicator dynamics of a CDA game in Market 3. Here, we have an attractor equilibria at A=(1, 0, 0) and C=(0, 0.29, 0.71) and a saddle point at D=(0, 0.51, 0.49). Note that the magnitude of the replicator dynamics is very small compared to that of Market 1.

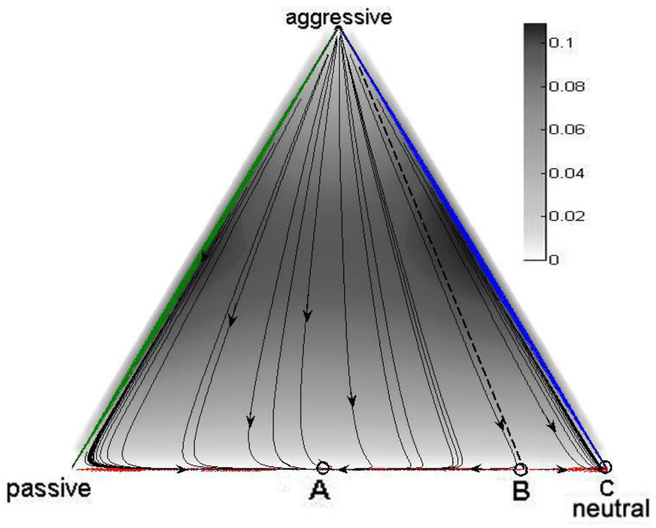


Fig. 6. The replicator dynamics of a CDA game and market shock *MS12*. Here, we have an attractor equilibrium at A=(0.56, 0.44, 0) and C=(0, 1, 0), and a saddle point at B=(0.16, 0.84, 0).

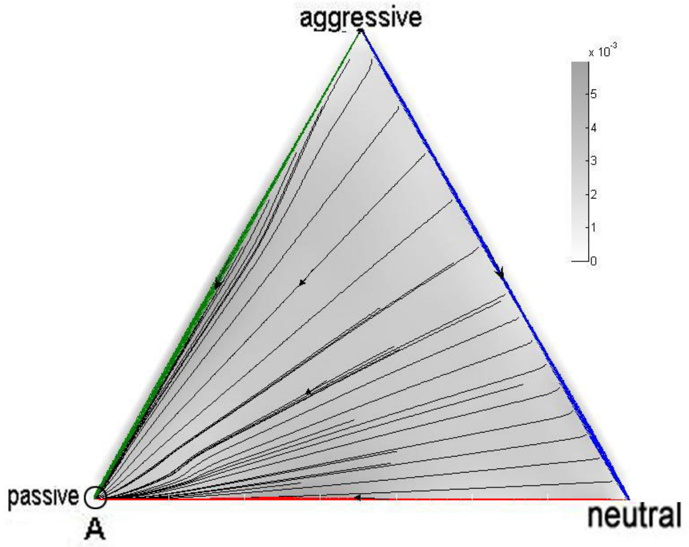


Fig. 7. The replicator dynamics of a CDA game and market shock *MS13*. Here, we have an attractor equilibrium at A=$(1, 0, 0)$.

With market shock *MS13*, we have a single attractor at A. Here, all the agents very slowly, but eventually, adopt passive behaviour. However, there is comparatively little to be gained in moving towards the evolutionary stable passive behaviour (for the same reasons we outlined in our discussion of Market 3).

6 Conclusions and Future Work

As electronic marketplaces become ever more common, we believe software agents will increasingly come to dominate the trading landscape. Their ability to quickly make informed decisions, based on the available data, make them ideal candidates for automated trading. To this end, analysing the impact of varying one of the fundamental characteristics of their bidding behaviour in a range of market situations is an important step. In particular, in this paper, we show that in a symmetric market, an agent is more likely to adopt an evolutionary stable neutral behaviour. However, when there are market shocks that increase the equilibrium price but maintain the symmetry of the market (meaning agents have to update their beliefs of the market) neutral is no longer the behaviour agents are most likely to adopt. In this case, more agents change to being passive. We also observe that changing behaviour is not particularly profitable for an agent in an asymmetric market.

For future work, we intend to look at other types of symmetric and asymmetric demand and supply, and other types of market shock in order to obtain further insights into how a trader's behaviour changes in yet other types of market.

For completeness, we also aim to address the limitation of our model where an agent has the same behavioural type when it is both a buyer and a seller. In particular, we believe that separately analysing these two roles can be more insightful, particular in asymmetric markets, where we do not expect the same behaviour from them.

Acknowledgments

We kindly thank anonymous reviewers for their helpful comments on the paper. The first author is funded by the DIF-DTC 8.6 project (www.difdtc.com).

References

1. D. Abreu and R. Sethi. Evolutionary stability in reputational model of bargaining. *Games and Economic Behavior*, 44(2):195–216, 2003.
2. A. Byde. Applying evolutionary game theory to auction mechanism design. *ACM Conference on Electronic Commerce*, pages 192–198, 2003.
3. D. Cliff. ZIP60: Further explorations in the evolutionary design of online auction market mechanisms. Technical Report HPL-2005-85, 2005.
4. D. Friedman and J. Rust. *The Double Auction Market: Institutions, Theories and Evidence*. Addison-Wesley, New York, 1992.
5. S. Gjerstad and J. Dickhaut. Price formation in double auctions. *Games and Economic Behavior*, 22:1–29, 1998.
6. M. He, N. R. Jennings, and H. Leung. On agent-mediated electronic commerce. *IEEE Trans on Knowledge and Data Engineering*, 15(4):985–1003, 2003.
7. M. Hollander and D. A. Wolfe. *Nonparametric Statistical Methods*. Wiley, 1973.
8. R. D. McKelvey and A. McLennan. Computation of equilibria in finite games. *Handbook of Computational Economics*, 1, 1996.
9. S. Phelps, S. Parsons, and P. McBurney. An evolutionary game-theoretic comparision of two double auction market designs. *Proceedings of the 6th Workshop on Agent Mediated Electronic Commerce*, pages 192–198, 2004.
10. V. L. Smith. An experimental study of competitive market behaviour. *Journal of Political Economy*, 70:111–137, 1962.
11. G. Tesauro and J. L. Bredin. Strategic sequential bidding in auctions using dynamic programming. *Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, pages 591–598, 2002.
12. G. Tesauro and R. Das. High-performance bidding agents for the continuous double auction. *Proceedings of the Third ACM Conference on Electronic Commerce*, pages 206–209, 2001.
13. K. Tuyls, T. Lenaerts, K. Verbeeck, S. Maes, and B. Manderick. Towards a relation between learning agents and evolutionary dynamics. *Proceedings of BNAIC 2002*, pages 21–22, 2002.
14. P. Vytelingum, R. K. Dash, E. David, and N. R. Jennings. A risk-based bidding strategy for continuous double auctions. *Proc. 16th European Conference on Artificial Intelligence*, pages 79–83, 2004.

15. W. Walsh, D. Parkes, and R. Das. Choosing samples to compute heuristic-strategy nash equilibrium. *Proceedings of the Fifth Workshop on Agent-Mediated Electronic Commerce*, 2003.
16. W. E. Walsh, R. Das, G. Tesauro, and J. O. Kephart. Analyzing complex strategic interactions in multi-agent games. *Proceedings of AAAI-02 Workshop on Game-Theoretic and Decision-Theoretic Agents (GTDT-02)*, 2002.
17. J. W. Weibull. *Evolutionary Game Theory*. MIT Press, Cambridge, MA, 1995.

A Fast Method for Learning Non-linear Preferences Online Using Anonymous Negotiation Data

D.J.A. Somefun and J.A. La Poutré

Center for Mathematics and Computer Science (CWI), P.O. Box 94079, 1090 GB Amsterdam, The Netherlands
{koye,hlp}@cwi.nl

Abstract. In this paper, we consider the problem of a shop agent negotiating *bilaterally* with *many* customers about a bundle of goods or services together with a price. To facilitate the shop agent's search for mutually beneficial alternative bundles, we develop a method for online learning customers' preferences, while respecting their privacy. By introducing additional parameters, we represent customers' highly nonlinear preferences as a linear model. We develop a method for learning the underlying stochastic process of these parameters online. As the conducted computer experiments show, the developed method has a number of advantages: it scales well, the acquired knowledge is robust towards changes in the shop's pricing strategy, and it performs well even if customers behave strategically.

1 Introduction

Combining two or more items and selling them as one good, a practice called bundling, can be a very effective strategy for reducing the costs of producing, marketing, and selling products. In addition, and maybe more importantly, bundling can stimulate demand for (other) goods or services [1]. To stimulate demand by offering bundles of goods, requires knowledge of customer preferences. Traditionally, firms first acquire such aggregate knowledge about customer preferences, for example through market research or sales data, and then use this knowledge to determine which bundle-price combinations they should offer. Especially for online shops, an appealing alternative approach would be to *negotiate* bundle-price combinations with customers: in that case, aggregate knowledge can be used to facilitate an interactive search for the desired bundle and price. Due to the inherently interactive characteristics of negotiation, such an approach can very effectively adapt the configuration of a bundle to the preferences of a customer. A high degree of bundle customization can increase customer satisfaction, which may lead to an increase in the demand for future goods or services.

In this paper, we consider the problem of a shop agent negotiating *bilaterally* with *many* customers about selecting a subset from a collection of goods or services, viz. the bundle, together with a price for that bundle. Thus, the bundle configuration—an array of bits, representing the presence or absence of each of the shop's goods and services in the bundle—together with a price for the bundle, form the negotiation issues. Like the work of [5,6,3,9,2,7], the techniques developed in this paper try to benefit from the so-called win-win opportunities, by finding mutually beneficial alternative bundles during negotiations.

The methods proposed in Faratin et al. [3], Coehoorn and Jennings [2] are geared towards finding win-win opportunities through modeling the preferences of the negotiation partner, for issues with independent valuations. This paper follows a similar approach; like our earlier work [9,7] and Klein et al. [6], we consider *interdependencies* between issues, however, which make the problem considerably harder. The novelty of our current approach lies in the introduction of a new scalable and very fast method for *online* learning these complex customer preferences, while respecting their privacy. To respect customers' privacy, the method as developed only uses *anonymous* data on passed negotiations: i.e., the shop cannot determine whether he dealt with a customer in the past. As the conducted computer experiments show, the necessary aggregate information, to significantly facilitate the search for mutually beneficial alternative bundles in future bilateral negotiations, is learned quickly for relatively large problems.

The next Section provides a high-level overview of the interaction between shop and a customer; moreover it specifies the recommendation mechanism (which is a simplified version of the mechanism we developed in [9]). In order for this recommendation mechanism to work well, it requires aggregate (anonymous) knowledge about customer preferences. Section 3 discusses the learning problem of obtaining the necessary knowledge. Section 4 introduces the online method for solving the learning problem. Section 5 presents the conducted computer experiments and discusses the results. Conclusions follow in Section 6.

2 Negotiation Process

Section 2.1 gives an overview of the interaction between the shop and a customer, as they try to negotiate an agreement about the price and the composition of a bundle of goods. Section 2.2, describes a simplified version of the advising mechanism we introduced earlier (cf. [9]). The shop uses this advising mechanism to suggest alternative bundles to his customers. As an input, this advising mechanism requires knowledge of how customer preferences are distributed. Section 4 introduces an online mechanism for learning this distribution. Note that the shop cannot use these two mechanisms simultaneously; per customer he decides whether to exploit his current knowledge (i.e., use the advising mechanism) or explore a bit about the current customer's preferences (i.e., use the online learning mechanism to update his knowledge).

2.1 Interaction Shop and Customer

The shop sells a total of n goods, each of which may be either absent or present in a bundle, so that there are $2^n - 1$ distinct bundle-configurations containing at least 1 good. In the current paper, we use $n = 20$. A negotiation concerns a

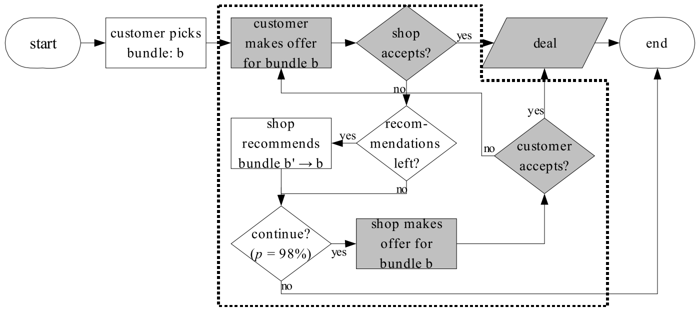


Fig. 1. A flowchart giving a high level overview of the interaction

bundle (configuration), together with a price for that bundle, and it is conducted in an alternating exchange of offers and counter offers [8], typically initiated by a customer. An example of such a practice may involve the sales of bundles of news items in categories like politics, finance, economy, sports, arts, etc.

During the negotiation process, the shop will suggest (a limited number of) alternative bundles. The hope is that these suggestions will lead to the bargainers finding mutually beneficial alternative bundles. More technically, such a mutually beneficial alternative bundle represents a *Pareto improvement* because switching to that bundle makes one bargainer better off without making the other worse off. Ideally, the shop will eventually find a bundle for which no more Pareto improvements exits: such a bundle is called a *Pareto efficient* bundle. To ease the description of our model and solutions, customers specify the bundle content for the opening offer, and thereafter only the shop can change the bundle content of an offer. The possibility of customers explicitly rejecting or changing the bundle content can be easily incorporated in our model and solutions, however.

Figure 1 provides a high-level overview of the interaction between a shop and a customer. The shaded elements are part of the actual negotiation—the exchange of offers. The process starts with the customer indicating her interests, by specifying $\mathbf{b}$, the bundle they will initially negotiate over. After that, they enter into a loop (indicated by the dotted line) which ends only when a deal is made, or with a 2% exogenous probability. (We do not model bargainers' impatience explicitly; therefore we need an exogenous stopping condition, which specifies the chance of bargaining breakdown.) In the loop, customer and shop agent negotiate in an alternating exchange of offers and counter offers. At the beginning of the negotiation the shop will construct an "advice set," which contains all the bundles that lie in the neighborhood of the initial bundle $\mathbf{b}$; the shop agent will use this set to search for Pareto improvements. Whenever it is the shop agent's turn, it removes the most promising bundles, say $\mathbf{b}'$, form the advice set and submits a counter offer containing the most promising bundle, $\mathbf{b}'$. Whenever the advice set turns out empty, the shop agent will thereafter only negotiate over the bundle that in the performed local search turned out to be the "best bundle." Moreover, under certain conditions bundles may also be added to the advice set. Section 2.2 gives the details of the advising mechanism.

2.2 Advising Mechanism

The idea is to device a mechanism that tells the seller what to recommend given the current state of the ongoing negotiation process. Ideally, this mechanism will lead to the shop and a customer ending up negotiating over a Pareto efficient bundle. Whether a bundle is Pareto efficient depends on the shop and customer's preferences. We assume that all customers and the shop order bundles based on their *net monetary value*; the bundle with the highest net monetary value is the most preferred bundle. A customer's net monetary value of a bundle is equal to the customer's valuation of the bundle (expressed in money) minus the bundle price and the shop's net monetary value is equal to the bundle price minus the shop's bundle valuation (also expressed in money). In this situation, the *gains from trade* are defined as the difference between the customer and seller's valuation.

Given the above assumption and the assumption that a customer wants to buy at most one bundle (within a given time period), it is straightforward to show that a bundle is Pareto efficient if and only if it results in the highest gains from trade (cf. [9] for a formal proof). Suppose customers' valuation for a bundle is randomly distributed and more importantly, the shop has learned this distribution. Then faced with the problem of recommending one bundle out of a collection of bundles, the shop will recommend the bundle that is most likely to result in higher gains from trade.

A customer initiates the negotiation process by proposing an initial bundle $\mathbf{b}$ and offering an opening price. The shop stores the bundle proposed by the customer as (his assessment or estimation of) the customer's "interest bundle," in the neighborhood of which the shop searches for promising alternative bundles to recommend. This neighborhood of bundle $\mathbf{b}$, $Ng(\mathbf{b})$, is defined as the bundles which, in binary representation, have a Hamming distance to $\mathbf{b}$ of 1. The advantage of advising bundles within the neighborhood of $\mathbf{b}$ is that the advice is less likely to appear haphazard.

Having defined a bundle's neighborhood, let the ordered set Adv denote the so-called "recommendation set," obtained by ordering the neighborhood $Ng(\mathbf{b})$ on the basis of the estimated probability that switching from bundle $\mathbf{b}$ to a bundle $\mathbf{b}'$ within $\mathbf{b}$'s neighborhood result in higher gains from trade.

To recommend a bundle $\mathbf{b}_k$ (the k^{th} recommendation, with $k \geq 1$), the mechanism removes the first bundle from Adv, adds a price to it and proposes it as part of the shop's next offer. Depending on the customer's counter offer for bundle $\mathbf{b}_k$, the current advice set may be replaced: if the customer's response is very promising (to be defined below) Adv will be emptied, bundle $\mathbf{b}_k$ will be taken as the customer's *new* interest bundle (in the neighborhood of which the search continues), and the bundles in the neighborhood of $\mathbf{b}_k$ are added to Adv.

A bundle $\mathbf{b}_k$ is very promising, if its estimated gains from trade—defined as customers current price minus the shop's reservation value for that bundle—exceed the highest estimated gains from trade for the current interest bundle. In that case, the shop's current assessment of the customer's interest bundle is updated to $\mathbf{b}_k$: the customer's response is promising enough to divert the search towards the neighborhood of $\mathbf{b}_k$. That is, the first element of *Adv* becomes the bundle with the highest estimated probability that switching to this bundle results in higher gains from trade, the second element of *Adv* becomes the bundle with the second highest estimated probability, and so on.

In case customer's response is not promising enough, the shop will make the next recommendation. Before the shop makes the next recommendation however, he checks if the negotiation is currently about the interest bundle. If this is not the case, he will first make an offer containing the interest bundle. Whenever the customer does not accept this offer, the shop will make the next recommendation in the following round. Consequently, we have the property that a recommendation is always preceded by an offer containing the interest bundle. (That is, the best estimation of the interest bundle's gains from trade is updated frequently, making a change of interest bundle, merely because the customer uses an ascending offer strategy, less likely.)

The search for Pareto improvements terminates whenever all the bundles in the search set have been considered and none turned out promising enough. After the search ends the shop agent continues negotiating about the interest bundle (which is most likely to represent a Pareto efficient bundle) until a deal is reached.

3 Problem Statement

3.1 Customer Preferences

There are n individual goods from which customers can compile the bundle they wish to purchase. The column vector $\mathbf{b} = [b(1), \ldots, b(n)]^T$ denotes the binary representation of a bundle (i.e., $b(i) = 1$ if and only if good i is part of the bundle). Due to the possibilities of synergy effects (either positive or negative), it does not suffices to compute customer c's monetary valuation for bundle $\mathbf{b}$ by just adding up the value of the individual goods. These synergy effects may occur because two, three, up to n goods are bought at the same time. In other words, a customer's monetary valuation for bundle $\mathbf{b}$, is obtained by adding up the values for all individual goods that constitute the bundle $\mathbf{b}$ and the synergy effects of all possible subsets of bundle $\mathbf{b}$, with more than one good.

More formally, let a_i denote a customer's valuation for obtaining good i individually and $a_{i,j}$, $a_{i,j,l}$, etc denote the synergy effects exclusively caused by obtaining a subset of goods $\{i, j\}$, $\{i, j, l\}$, etc: e.g., a_{ij} is equal to the customer's valuation for bundle $\{i, j\}$ minus the valuation for obtaining the goods individually. There are $\binom{n}{1}$ parameters of the type a_i, $\binom{n}{2}$ parameters of the type a_{ij}, and so on; thus there are in total at most $k = \sum_{i=1}^{n} \binom{n}{i} = 2^n - 1$ parameters. Clearly, dealing with $2^n - 1$ parameters becomes already impractical whenever n is not a relatively small number, say $n > 10$. In the experiments, we will consider problems where $n = 20$. To make the number of parameters manageable we will then assume that beyond subsets of size 3 the synergy effect are always zero: i.e., there are only $k = \binom{n}{1} + \binom{n}{2} + \binom{n}{3}$ number of parameters. (This does not mean, the absence of nonlinearities for larger sized bundle; these nonlinearities are just caused by lower ordered synergy effects.) For now it suffices to assume that k has a manageable size.

Given some (arbitrary but) fixed encoding, denoted by $< \cdot >$, we can map all these k-parameters into the (k-dimensional) vector $\mathbf{a}$. We can now define the customer's monetary valuation for bundle $\mathbf{b}$, $v^c_{\mathbf{a}}(\mathbf{b})$, as follows:

$$\begin{aligned} v^c_{\mathbf{a}}(\mathbf{b}) = & \sum_{i=1}^{n} \mathbf{a}_{<i>} b(i) + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \mathbf{a}_{<i,j>} b(i) b(j) + \ldots \\ & + \mathbf{a}_{<1,\ldots,n>} b(1) b(2) \ldots b(n). \end{aligned} \tag{1}$$

3.2 Learning Problem

Based on Eq. (1) and the encoding $< \cdot >$, we can conclude that there exist a mapping $\mathbf{f} : \{0,1\}^n \mapsto \{0,1\}^k$—which is known by the seller—such that for all customers, their monetary valuation for any bundle can be expressed as a linear function of the values of their k-parameters: i.e., ${v_{\mathbf{a}}}^c(\mathbf{b}) = \mathbf{f}(\mathbf{b})^T \mathbf{a}$, for any bundle $\mathbf{b}$ and $f(\mathbf{b})^T$ is the transpose of the k dimensional column vector generated by $\mathbf{f}(\mathbf{b})$. *For example*, if there are only 3 goods—good 1,2, and 3—then $\mathbf{b} = (1,1,0)$ could (dependable on the encoding) give $\mathbf{f}(\mathbf{b}) = (1,1,0,1,0,0,0)$. Irrespective of the encoding, $\mathbf{f}(1,1,0)$ should contain three 1's and four 0's: i.e. the parameters a_1, a_2, and $a_{<1,2>}$ contribute to the valuation of bundle $(1,1,0)$.

In the absence of strategic behavior and no deviations in customer preferences, $\mathbf{a}$ is the same for all customers. The seller can then just obtain $\mathbf{a}$ by solving a system of linear equations. For example, suppose for $i = 1,\ldots,t$ the seller has the observations v_i and $\mathbf{b}_i$, where v_i denotes customers' valuations for bundle $\mathbf{b}_i$. Then

$$\mathbf{v} = \mathbf{B}\mathbf{a} \tag{2}$$

with $\mathbf{v} = [v_1,\ldots,v_t]^T$ and $\mathbf{B} = [\mathbf{f}(\mathbf{b}_1),\ldots,\mathbf{f}(\mathbf{b}_t)]^T$ (thus $\mathbf{B}$ is a $t \times k$ matrix). Since the seller also knows $\mathbf{f}$, he knows both $\mathbf{v}$ and $\mathbf{B}$. Now if the rank of matrix $\mathbf{B}$ is equal to k, then the (Moore-Penrose) generalized matrix inverse of $\mathbf{B}$, denoted by $\mathbf{B}^+$, can be used to compute $\mathbf{a}$: i.e., $\mathbf{a} = \mathbf{B}^+\mathbf{v}$, where $\mathbf{B}^+\mathbf{v}$ is the least square (LS) solution of the systems of linear equations described by Eq.(2) (cf. [4]).

Generally, customer preferences differ from one customer to the next; consequently, the vector $\mathbf{a}$, which completely describes customer's preferences, will also differ. That is, $\mathbf{a}$ becomes a random vector; we assume it is normally distributed with a vector of means $\mu_{\mathbf{a}}$ and covariance matrix $\mathbf{\Sigma_a}$. In that case, Eq. (2) becomes

$$\mathbf{v} = \mathbf{B}\mu_{\mathbf{a}} + \epsilon, \tag{3}$$

with $\epsilon = [\epsilon_1,\ldots,\epsilon_t]^T$ and ϵ_i denotes the disturbance at observation i; ϵ_i is normally distributed with mean zero and variance $\mathbf{f}(\mathbf{b})^T \mathbf{\Sigma_a} \mathbf{f}(\mathbf{b})$ (for $i = 1,\ldots,t$). Thus the disturbance variance is not a constant across observations: i.e., heteroscedasticity occurs. Consequently, we can now no longer apply LS in order to find a "good" estimation of $\mu_{\mathbf{a}}$. (By a good estimation, we will just mean maximum likelihood estimation.)

Performing simple LS in case of hetroscedasticity means among other things that you will not obtain a maximum likelihood estimation of $\mu_{\mathbf{a}}$ (cf. [4] for a detailed discussion). With hetroscedasticity, one often resorts to general least square (GLS), but then we at least need to have some idea of how the various disturbances are related. In this case, the disturbance ϵ_i depends on the bundle content x_i, which is observed. We may be able to use this knowledge to design a GLS estimator. Important drawbacks are however that this GLS estimator cannot be performed online and it does not scale well: in total 2^n different bundle contents are possible, hence knowing that there is a relation between bundle content and disturbance is not very helpful for large n's.

Remember, the shop negotiates with customers over the various bundle and price combinations. Consequently, the shop can influence the quality of the *anonymous* data on past negotiations. We will show that by a clever manipulation of this data the shop can reduce the problem specified in Eq. (3) to a straightforward maximum likelihood problem, avoiding hetroscedasticity altogether. In the absence of strategic behavior we get very accurate estimations of $\mu_{\mathbf{a}}$ and $\mathbf{\Sigma}_{\mathbf{a}}$ with this procedure. Whenever customers behave strategically this is generally not possible, however.

With strategic behavior, the seller does not directly observe $\mathbf{v}$. Instead, he observes $\mathbf{p} = [p_1, \dots, p_t]$ which denotes the sequence of bids submitted by the customers. We have that

$$\mathbf{p} = \mathbf{v} + \mathbf{s}, \tag{4}$$

where $\mathbf{s} = [s_1, \dots, s_t]^T$ and s_i (for $i = 1, \dots, t$) denotes the extra disturbance caused by strategic behavior. Generally, the disturbance s_i has a mean greater or equal than zero. We will show that by a clever preprocessing of the data $\mathbf{p}$, the shop can nevertheless eradicate a sufficient amount of the noise caused by strategic behavior. As a result, the shop can learn enough about customers' preferences to facilitate the search for Pareto efficient bundles, significantly.

4 Learning Method

The method we develop, for learning (more about) customer preferences, contains (a) a sub method for learning customer preferences in the absence of strategic behavior and (b) a sub method for preprocessing the negotiation data. First method (b) is applied to the data such that a sufficient amount of the noise caused by strategic behavior is removed. Next method (a) is applied to the preprocessed data. In Section 4.1 and 4.2 we discuss method (a) and (b), respectively.

4.1 Estimating Mean and Covariance Matrix

As was discussed in Section 3, per customer we have an instance of the random vector $\mathbf{a}$, which completely specifies her preferences. For a customer c, let $\mathbf{a}_c$ denote such an instance. Suppose the seller would negotiate for a very long time with this customer c, while constantly suggesting different bundles. In the absence of strategic behavior, eventually the seller would have collected the negotiation data $\mathbf{v}_c = [v_{c1}, \ldots, v_{ct}]^T$, and $\mathbf{B}_c = [\mathbf{f}(\mathbf{b}_{c1}), \ldots, \mathbf{f}(\mathbf{b}_{ct})]^T$ such that $\mathbf{B}_c$ has rank k. Analogue to Eq. (2) (see Section 3.2), the seller can then just obtain $\mathbf{a}_c$ by solving $\mathbf{v}_c = \mathbf{B}_c\mathbf{a}_c$: i.e., $\mathbf{a_c} = \mathbf{B}_c^+\mathbf{v}_c$, where $\mathbf{B}_c^+$ is again the generalized matrix inverse of $\mathbf{B}_c$ and $\mathbf{B}_c^+\mathbf{v}_c$ is the LS solution of the linear system.

It becomes rather straightforward to obtain good estimations of $\mathbf{a}$'s vector of means ($\mu_\mathbf{a}$) and covariance matrix ($\mathbf{\Sigma_a}$), if the seller could directly obtain the vector $\mathbf{a_c}$ from all or most customers he negotiates with (by just computing $\mathbf{a_c} = \mathbf{B}_c^+\mathbf{v}$). Clearly, for large k bargainers will break off the negotiations or reach a deal, before the matrix $\mathbf{B}_c$, grows into a rank k matrix. Consequently, it is very unlikely that the seller will be able to obtain $\mathbf{a_c}$ for even one customer. Nevertheless, the seller can use this idea of obtaining a solvable system of linear equations to at least partly reveal $\mathbf{a}_c$, for most customers.

To obtain such a solvable system, the seller suggests a sequence of bundles such that the union of the goods contained in these bundles has strictly less goods than is possible. (In the experiments, 4, 5, or at most 6 individual goods are being considered.) In that case, relative to k the matrix $\mathbf{B}_c$ contains only a few nonzero columns. Whenever k', the rank of the matrix $\mathbf{B}_c$, equals the number of nonzero columns, we can obtain $\mathbf{a}'_c$—the k' dimensional vector containing all the identifiable parameters of $\mathbf{a}_c$—by solving $\mathbf{v}_c = \mathbf{B}'_c\mathbf{a}'_c$ (i.e., $\mathbf{a}'_c = \mathbf{B}'^+_c\mathbf{v}_b$); we get the matrix $\mathbf{B}'_c$ by removing all zero columns from $\mathbf{B}_c$. Alg. (1) gives the pseudo code for computing $\mathbf{a}'_c$. Based on these newly computed parameter values, Alg. (2) updates their sum and sum of squares. It is now rather straightforward to obtain estimations of $\mu_\mathbf{a}$, the mean, and $\mathbf{\Sigma_a}$, the covariance matrix (see also the comment on line 9 and 10 of Alg. (2)).

Algorithm 1. Pseudo code for computing $\mathbf{a}'_c$, the values of the identifiable parameters of $\mathbf{a}_c$, and A, the ordered set such that its ith element ($A(i)$) identifies the element in $\mathbf{a}_c$ to which $\mathbf{a}'_c(i)$ corresponds.

```
After $t$ consecutive rounds negotiating with customer $c$, we have $\mathbf{B}_c$ and $\mathbf{v}_c$.
1. $A := \emptyset$
2. For $(1 \leq i \leq t)$
3.   if (column $i$ of $B$ does not only contain 0's)$\{A := A \cup i\}$
4. For $(1 \leq i \leq t)$
5.   if ($i$ not in $A$)$\{$remove column $i$ from $B\}$
6. if (rank $\mathbf{B}_c$ equals $|A|$)
7.   $\mathbf{a}'_c := \mathbf{B}_c^+\mathbf{v}$ //$\mathbf{B}_c^+$ is the generalized matrix inverse of $\mathbf{B}_c$
8. return $\mathbf{a}'_c$ and $A$
```

Algorithm 2. Pseudo code for updating the sum and sum of squares of the identifiable parameter values of $\mathbf{a}$; based on these values we can easily compute μ_a^e and $\mathbf{\Sigma}_a^e$ the current estimation of the mean ($\mu_{\mathbf{a}}$) and covariance matrix ($\mathbf{\Sigma_a}$).

```
Given is $\mathbf{a}'_c$ and $A$.
1. For $(1 \leq i \leq |A|)\{$
2.   $i' := A(i) // A(i)$ is the $i$th element of $A$
3.   $sum_{a_{i'}} := sum_{a_{i'}} + \mathbf{a}'_c(i)$
4.   $sum_{i'} := sum_{i'} + 1$
5.   For $(i \leq j \leq |A|)\{$
6.      $j' := A(j) // A(j)$ is the $j$th element of $A$
7.      $sumSquare_{a_{i'}a_{j'}} := sumSquare_{a_{i'}a_{j'}} + \mathbf{a}'_c(i')\mathbf{a}'_c(j')$
8.      $sum_{i'j'} := sum_{i'j'} + 1\}$ $\}$
9.//Note, for $1 \leq i, j \leq k$, $\mu_{\mathbf{a}}^e(i) := sum_{a_i}/sum_i$ and
10.//$\mathbf{\Sigma}_{\mathbf{a}}^e(i,j) = \mathbf{\Sigma}_{\mathbf{a}}^e(j,i) := \frac{sumSquare_{a_i a_j}}{sum_{ij}} - \mu_{\mathbf{a}}^e(i)\mu_{\mathbf{a}}^e(j)$
```

4.2 Preprocessing Negotiation Data

With strategic behavior the seller does not observe $\mathbf{v}_c = [v_{c1}, \ldots, v_{ct}]^T$. Instead he observes $\mathbf{p}_c = [p_{c1}, \ldots, p_{ct}]$, the sequence of bids submitted by the customers, where there is generally a difference between p_{ci} and v_{ci} (see also Eq. (4)). To deal with strategic behavior requires some implicit or explicit model of customers' bidding strategy. For example, in [9,2] they learn more about customer preferences based on the difference between two consecutive bids. This implies some implicit model of a customer's bidding behavior, however. We want to make this model explicit such that we can check whether the model is consistent with a customer's bidding behavior.

Strategic behavior of customers can roughly be split into a fixed and variable component. The fixed component specifies how much a customer c is at most willing to bid for a bundle $\mathbf{b}$ worth $\mathbf{f}(\mathbf{b})^T\mathbf{a}_c$, irrespective of the current state of the negotiation; the variable component specifies how a customer concedes through time. The seller can use the gathered negotiation data with customer c ($\mathbf{B}_c$ and $\mathbf{p}_c$) to eradicated as much noise as possible.

The idea is that at the end of a negotiation process the seller will use some of the gathered negotiation data to model how the opponent concedes trough time; the rest of the data can then be used to check whether the developed model correctly predicts how a customer concedes through time. It is very unlikely that such a test is passed incorrectly (error of the first kind) because both the correctness of the estimated values for the identifiable parameters ($\mathbf{a}'_c$) and the concession model are tested jointly. Moreover, if by chance this type of error occurs the resulting noise in the estimated values of $\mathbf{a}'_c$ will generally be unbiased. The other possibility, of incorrectly rejecting all available models for a particular negotiation is even less problematic. The seller can just simple ignore this negotiation data. In real world applications the seller could have a collection of typical models and could check which of the models fits the data of a particular negotiation best. Consequently, enough negotiation data can be preprocessed and fed to the online learning algorithm (developed in Section 4.1) to ensure accurate estimation of the $\mathbf{a}$'s vector of means and covariance matrix.

Filtering out the fixed component of a customer's bargaining strategy is impossible. Moreover, this type of behavior leads to biased estimation errors. In the conducted experiments we therefore focus on assessing the robustness of the developed advising mechanism to inaccuracies in the estimated vector of means ($\mu_{\mathbf{a}}$) and covariance matrix ($\mathbf{\Sigma_a}$), due to this type of strategic behavior. To further streamline the experiments, the seller uses a concession model of the opponent, which (given all data available at termination of a negotiation) is accurate enough to eradicate most of the noise caused by a customer conceding through time. (Due to space limitations, we have to leave out the details, however.)

5 Numerical Experiments

5.1 Experimental Design

Customer Preferences Revisited. In Section 3.1 we already discussed the general preference model. In the experiments, we consider the highly nonlinear case of (direct) synergy effect between subsets of size 3 and less. That is, the k-dimensional random vector $\mathbf{a}$ contains $k = \binom{n}{1} + \binom{n}{2} + \binom{n}{3}$ parameters. In the experiments $n = 20$ (i.e., there are 20 individual goods) thus $k = 1350$. For a customer c, $\mathbf{a}_c$ is randomly drawn from the normal distribution $N[\mu_{\mathbf{a}}, \mathbf{\Sigma_a}]$. The vector $\mu_{\mathbf{a}}$ and the matrix $\mathbf{\Sigma_a} = [\sigma_a(i,j)]$ denote the means and (co)variances of the distribution. We assume that the parameters are uncorrelated: i.e., $\sigma_a(i,j) = 0$ for all $i \neq j$.

Modeling Negotiations. In the experiments, we assume that a customer is at most willing to pay x% of their true valuation for the queried bundles; for each customer x is randomly drawn between 85% and 95%. The shop is willing to drop to the reservation value. Moreover, both customer and shop use time-dependent bidding strategies: the offer submitted by the customer (shop) is monotonically increasing (decreasing) in both the number of bidding rounds and the valuation. More specifically, a bidding strategy is characterized by the gap the customer leaves between the initial offer and the maximum price, and by the speed with which it closes this gap: i.e., $p(t) = (1 - gap(t))x \cdot v(\mathbf{b})$, where $x \cdot v(\mathbf{b})$ denotes the maximum price a customer is willing to pay for bundle $\mathbf{b}$. The gap is specified as a fraction of the maximum price and it decreases over time as $gap(t) = gap_{init} \cdot \exp(-\delta t)$, so over time, the customer's bids approach the maximum price of the bundle it is currently negotiating over. Note that changes in the gap are time-dependent, but not bundle-dependent! Almost the same holds for the shop's bidding strategy, *mutatis mutandis*. The initial gap, gap_{init}, is uniformly drawn between 0.4 and 0.6, for the proxy agent and it is set at -0.5 for the shop. Moreover we uniformly draw δ between 0.02 and 0.04 for the proxy agent and δ is fixed to 0.03 for the shop.

Experimental Setup. In the computer experiments reported in this paper, we compare our new approach of having *no a priori information* and learning customer preferences online (as discussed in Section 4), to the one where—for example because of expert knowledge—the shop already knows the underlying joint probability distribution of all bundle valuations (see Section 3.1 and 5.1). Besides comparing our new online procedure (referred to as ML) with the above method (called S), we also assess the relative performance of the system by performing the same series of experiments with a benchmark procedure (called B), which simply recommends a random bundle from the current bundle's neighborhood. That is, the benchmark does not base the order in which it recommends the next bundle on the estimated likelihood of higher gains from trade, as our system does.

In the experiments, the shop's bundle valuations are determined by applying a nonlinear bundle reduction. This means that the bundle price is generally less than the sum of the individual goods comprising the bundle. In order to prevent the trivial problem of customers wanting to buy all goods, the bundle reduction becomes 0 for bundles which contain more than 3 goods. Moreover, whenever the shop employs the ML-system, he uses slightly different settings for δ and gap_{init} in the exploration phase: i.e., $\delta = 0.015$ and $gap_{init} = -3.0$. Thus, in order to allow for enough search, the shop has a higher initial ask price and concedes slower through time, in the exploration phase of the ML-system.

There are 20 individual goods. To mirror more realistic settings where there are only synergy effect for a limited number of goods, we randomly assign 80 two or three size bundles to have synergy effects. All the other nonlinear parameters have zero mean and variance. We randomly generate the means of all the nonzero parameters in two distinct ways: the mean of parameters referring to individual goods are uniformly drawn between 0 and 250; to allow for negative synergy effect, the mean of parameters referring to synergy effect are uniformly drawn between minus and plus 250. To ensure enough volatility, we set the variance of the parameters equal to their means. (Without volatility all that matters is the mean and the problem becomes rather easy.) To test the robustness of our procedure to quantitative changes in the underlying distributions, we conducted a series of experiments with 30 different distributions. For each of these settings we simulated negotiations between the shop, with randomly drawn valuations, which were kept constant across negotiations with $5,000$ customers, each with her valuations drawn randomly from the particular distribution used.

Customers initiate the negotiation by submitting an offer containing an initial bundle $\mathbf{b}_{init}$. To give the shop some room for improvement, we initialize the customer's initial bundle by randomly selecting a bundle $\mathbf{b}$ which, in binary representation, has a Hamming distance of 5 to the bundle $\mathbf{b}^*$ that is associated with the highest gains from trade.

5.2 Results

The overall results of our experiments are listed in Table 1. The numbers are averages over $5,000$ customers drawn from each distribution of valuations, and

Table 1. Comparison of the different methods ML, S, and B. Figures are averages across 30 runs with different random seeds, and 5,000 customers per run. Standard deviations are given between brackets.

	Performance							
Methods	max. gains	min. gains	gains $\mathbf{b}_{\text{init}}$	gains $\mathbf{b}_{\text{final}}$	perc.	rel. perc.	rounds	deals
ML				202.93	0.98	0.84	35.71	3134
				(121)	(0.00)	(0.02)	(10.6)	(436)
S	386.37	−8411.59	−891.27	245.97	0.98	0.88	35.01	3134
	(144.20)	(685.36)	(141.46)	(133)	(0.00)	(0.02)	(10.9)	(437)
B				−153.27	0.94	0.56	81.51	1884
				(72.20)	(0.01)	(0.04)	(12.08)	(381.7)

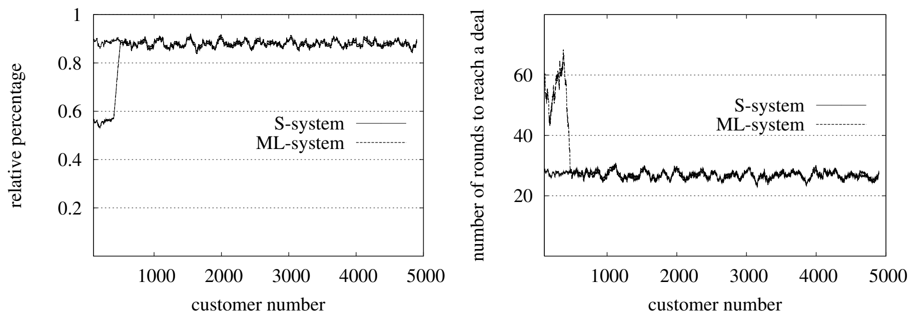


Fig. 2. Relative percentage and the number of rounds to reach a deal (on the left and right). A round constitutes an offer and accept/reject by the opponent. (Both graphs show the 100-step moving averages.)

over 30 different randomly generated distributions; standard deviations (across averages from the 30 distributions) are listed between brackets. The maximum and minimum attainable gains from trade are determined by the current random distribution of valuations; they do not depend on the chosen method. (We find these values by exhaustively searching the bundle space.) Likewise, the bundle a customer wants to start negotiating about does not depend on the chosen method. Therefore, the average of these figures represented in the first 3 columns are identical across all experiments—for each shop-customer interaction these figures are known even before the negotiation commences.

The remainder of the results is measured at the end of each shop-customer interaction, and subsequently averaged over all $30 \cdot 5,000$ customers. The column labeled 'gains $\mathbf{b}_{\text{final}}$' gives the gains from trade associated with the bundle that the shop and the customer were actually negotiating about at the end of the simulation, irrespective of whether that end was caused by the 98% exogenous break-up probability, or by the fact that a deal was reached in the negotiation. (The negative gains $\mathbf{b}_{\text{final}}$ for the B-system means that it often finds a bundle the customer is not willing to buy; this also partly explains the low average number of deals.) The columns for 'percentage' and 'rel(ative) percentage' present the same results in a different way: 'percentage' shows the shop's performance relative to the maximum attainable:

$$\text{percentage} = \frac{(\text{gains } \mathbf{b}_{\text{final}} - \text{min. gains})}{(\text{max. gains} - \text{min. gains})},$$

whereas 'relative percentage' takes into account the starting bundle b_{init}:

$$\text{relative percentage} = \frac{(\text{gains } \mathbf{b}_{\text{final}} - \text{gains } \mathbf{b}_{\text{init}})}{(\text{max. gains} - \text{gains } \mathbf{b}_{\text{init}})}.$$

In both these columns, as in all the columns more generally, the S-system only slightly outperforms the ML-system, which beats the B-system, but bear in mind the challenge for the ML-system, as compared to the S-system, in terms of (dealing with the lack of) available prior information about customer preferences. The columns labeled 'rounds' and 'deals' give the average number of rounds it took to reach a deal (whenever a deal was reached) and the average number of deals reached. Note the significant standard deviation of maximum and minimum attainable gains from trade. This indicate a significant difference between the 30 problem instances. Nevertheless, the standard deviations of key performance indicators are small, indicating that our results are relatively robust to changes in the problem instances.

To illustrate how well the ML-system performs, figure 2 shows the relative gains from trade and the number of rounds within which a deal is reached (if a deal is reached in the first place) per customer. These are two key performance indicators. To simplify the comparison we hard wired the ML-system to stop the exploration phase after 500 customers. (Please remember that learning is completely online, however.) In the exploitation phase of the ML-system, the two systems generate virtually the same performance. Figure 2 illustrates the great advantage of using negotiation data. In the exploration phase 500 customers suffices for the ML-system, because per customers the system obtains roughly 55 high quality data points. (55 is roughly the average number of rounds in the exploration phase.)

6 Conclusions

We consider the problem of a shop agent negotiating bilaterally with a customer about a bundle of goods or services together with a price. To facilitate the shop agent's search for mutually beneficial alternative bundles, we develop a method for online learning customers' preferences, while respecting their privacy. We conduct computer experiments with simulated customers that have highly *non-linear* preferences and are interested in bundles containing up to 20 individual goods.

We compare our new method of having *no a priori information* and learning about customer preferences online, to the one where—for example because of expert knowledge—the shop already knows the underlying joint probability distribution, according to which customers' preferences are generated. Both approaches use the same advising mechanism to search for mutually beneficial tradeoffs. Our experiments show how, after 500 customers, the two approaches have virtually the same performance. The learning method needs only 500 customers because it does not have to learn the underlying stochastic model completely. It suffices to learn enough to have the advising mechanism almost always suggest the most promising alternative bundle first, second most promising bundle second, and so on. Moreover, whenever the advising mechanism does make a mistake in the ordering it is robust enough to quickly correct the mistake, based on a customer's feedback.

Both methods significantly increase the speed with which deals are reached, as well as the number and the Pareto efficiency of the deals reached, as compared to a benchmark. Moreover, the experiments show that the developed learning method is able to learn the necessary information online, irrespective of the fact that due to strategic behavior customers' best offer is significantly les than their reservation value.

Acknowledgments

We would like to thank Tomas Klos; much of the presented work builds on earlier joined work with him.

References

1. Y. Bakos and E. Brynjolfsson. Bundling information goods: Pricing, profits and efficiency. *Management Science*, 45(12), December 1999.
2. R. M. Coehoorn and N. R. Jennings. Learning an opponent's preferences to make effective multi-issue negotiation tradeoffs. In *Proc. 6th Int Conf. on E-Commerce*, pages 59–68, 2004.
3. P. Faratin, C. Sierra, and N. R. Jennings. Using similarity criteria to make issue trade-offs. *Journal of Artificial Intelligence*, 142(2):205–237, 2003.
4. W. H. Greene. *Econometric Analysis.* Prentice-Hall, New Jersey, 1993.
5. C. Jonker and V. Robu. Automated multi-attribute negotiation with efficient use of incomplete preference information. In *3rd Int. Conf. on Autonomous Agents & Multi Agent Systems (AAMAS), New York*, pages 1056–1063, 2004.
6. M. Klein, P. Faratin, H. Sayama, and Y. Bar-Yam. Negotiating complex contracts. *Group Decision and Negotiation*, 12:111–125, 2003.
7. V. Robu, D. Somefun, and J. L. Poutré. Modeling complex multi-issue negotiations using utility graphs. In *the Fourth International Joint Conference on Autonomous Agents & Multi Agent Systems (AAMAS), Utrecht, July 25-29, 2005*, pages 280–287. ACM press, 2005.
8. A. Rubinstein. Perfect equilibrium in a bargaining model. *Econometrica*, 50(1): 97–109, January 1982.
9. D. J. A. Somefun, T. B. Klos, and J. A. La Poutré. Online learning of aggregate knowledge about nonlinear preferences applied to negotiating prices and bundles. In *Sixth International Conference on Electronic Commerce, ICEC04*, pages 361–370. ACM press, 2004.

Adaptive Pricing for Customers with Probabilistic Valuations

Michael Benisch, James Andrews, and Norman Sadeh

School of Computer Science, Carnegie Mellon University

Abstract. In this paper, we examine the problem of choosing discriminatory prices for customers with probabilistic valuations and a seller with indistinguishable copies of a good. We show that under certain assumptions this problem can be reduced to the continuous knapsack problem (CKP). We present a new fast ϵ-optimal algorithm for solving CKP instances with asymmetric concave reward functions. We also show that our algorithm can be extended beyond the CKP setting to handle pricing problems with overlapping goods (e.g.goods with common components or common resource requirements), rather than indistinguishable goods.

We provide a framework for learning distributions over customer valuations from historical data that are accurate and compatible with our CKP algorithm, and we validate our techniques with experiments on pricing instances derived from the Trading Agent Competition in Supply Chain Management (TAC SCM). Our results confirm that our algorithm converges to an ϵ-optimal solution more quickly in practice than an adaptation of a previously proposed greedy heuristic.

1 Introduction

In this paper we study a ubiquitous pricing problem: a seller with finite, indistinguishable copies of a good attempts to optimize profit in choosing discriminatory, take-it-or-leave-it offers for a set of customers. Each customer draws a valuation from some probability distribution known to the seller, and decides whether or not they will accept the seller's offers (we will refer to this as a *probabilistic pricing problem* for short). This setting characterizes existing electronic markets built around supply chains for goods or services. In such markets, sellers can build probabilistic valuation models for their customers, e.g.to capture uncertainty about prices offered by competitors, or to reflect the demand of their own customers.

We show that this pricing problem is equivalent to a continuous knapsack problem (CKP) (i. e. the pricing problem can be reduced to the knapsack problem and vice versa) under two reasonable assumptions: i.) that probabilistic demand is equivalent to actual demand, and ii.) that the seller does not wish to over promise goods in expectation. The CKP asks: given a knapsack with a weight limit and a set of weighted items – each with its value defined as a function of the fraction possessed – fill the knapsack with fractions of those items to maximize the knapsack's value. In the equivalent pricing problem, the items are the customer demand curves. The weight limit is the supply of the seller. The value of a fraction of an item is the expected value of that customer demand curve. The expected value is defined as the probability with which the customer is expected to accept the corresponding offer times the offer price.

Studies of CKPs in Artificial Intelligence (AI) and Operations Research (OR) most often focus on classes involving only linear and quadratic reward functions [11]. We present a fast algorithm for finding ϵ-optimal solutions to CKPs with arbitrary concave differentiable reward functions. The class of pricing problems that reduce to CKPs with such reward functions involve customers with valuation distributions that satisfy the diminishing returns (DMR) property. We further augment our CKP algorithm by providing a framework for learning accurate customer valuation distributions that satisfy this property from historical pricing data.

We also discuss extending our algorithm to solve pricing problems that involve sellers with distinguishable goods that require some indistinguishable shared resources (for example common components or shared assembly capacity). Such problems more accurately represent the movement from make-to-stock production to assemble-to-order and make-to-order production, but involve constraints that are too complex for traditional CKP algorithms.

The rest of this paper is structured as follows: In Section 2 we discuss related work on the probabilistic pricing and continuous knapsack problems. In Section 3 we present the pricing problem and its equivalence to continuous knapsack. In Section 4 we present our ϵ-optimal binary search algorithm for concave differentiable CKPs. Section 5 presents the framework for learning customer valuation functions. In Section 6 we validate our algorithm and framework empirically on instances derived from the Trading Agent Competition in Supply Chain Management (TAC SCM).

2 Background

2.1 Related Work on Pricing Problems

The pricing problem we study captures many real world settings, it is also the basis of interactions between customers and agents in the Trading Agent Competition in Supply Chain Management. TAC SCM is an international competition that revolves around a game featuring six competing agents each entered by a different team. In TAC SCM simulated customers submit requests for quotes (RFQs) which include a PC type, a quantity, a delivery date, a reserve price, and a tardiness penalty incurred for missing the requested delivery date. Agents can respond to RFQs with price quotes, or bids, and the agent that offers the lowest bid on an RFQ is rewarded with a contractual order (the reader is referred to [3] for the full game specification).

Other entrants from TAC SCM have published techniques that can be adapted to the setting we study. Pardoe and Stone proposed a heuristic algorithm with motivations similar to ours [9]. The algorithm greedily allocates resources to customers with the largest increase in price per additional unit sold. Benisch *et. al.*

suggested discretizing the space of prices and using Mixed Integer Programming to determine offers [1], however this technique requires a fairly coarse discretization on large-scale problems.

Sandholm and Suri provide research on the closely related setting of demand curve pricing. The work in [12] investigates the problem of a limited supply seller choosing discriminatory prices with respect to a set of demand curves. Under the assumptions we make, the optimal polynomial time pricing algorithm presented in [12] translates directly to the case when all customers have uniform valuation distributions. Additionally, the result that non-continuous demand functions are $\mathcal{NP}$-Complete to price optimally in [12], implies the same is true of non-continuous valuation distributions.

2.2 Related Work on Knapsack Problems

The traditional integer knapsack problem (where the amount of an item included in the knapsack must be an integer) has been well studied from an algorithmic perspective, and been shown to result from reductions of many types of problems in OR and AI [6]. There have been several algorithms developed for solving certain classes of continuous knapsack problems. When rewards are linear functions of the included fractions of items, it is well known that a greedy algorithm provides an optimal solution in polynomial time.[1] CKP instances with concave quadratic reward functions can be solved with standard quadratic programming solvers [11], or the algorithm provided by Sandholm and Suri. The only technique that generalizes beyond quadratic reward functions was presented by Melman and Rabinowitz in [8]. The technique in that paper provides a numerical solution to *symmetric* CKP instances where all reward functions are concave and identical.[2] However, this technique involves solving a difficult root finding problem, and its computational costs have not been fully explored.

2.3 Related Work on Learning Valuations

The third group of relevant work involves learning techniques for distributions over customer valuations. Relevant work on automated valuation profiling has focused primarily on first price sealed bid (FPSB) reverse auction settings. Reverse auctions refer to scenarios where several sellers are bidding for the business of a single customer. In the FPSB variant customers collect bids from all potential sellers and pay the price associated with the lowest bid to the lowest bidder. Predicting the winning bid in a first price reverse auction amounts to finding the largest price a seller could have offered the customer and still won. From the point of view of a seller, this price is equivalent to the customer's valuation for the good.

Pardoe and Stone provide a technique for learning distributions over FPSB reverse auctions in TAC SCM [9]. The technique involves discretizing the range

[1] Linear reward functions for CKP would result from a pricing problem where all customers have fixed valuations.

[2] Identical reward functions for CKP would result from a pricing problem where all customers draw valuations from the same distribution.

of possible customer valuations, and training a regression from historical data at each discrete valuation. The regression is used to predict the probability that a customer's valuation is less than or equal to the discrete point it is associated with. Similar techniques have been used to predict FPSB auction prices for IBM PCs [7], PDA's on eBay [5], and airline tickets [4].

3 Market Model

3.1 P3ID

We define the Probabilistic Pricing Problem with Indistinguishable Goods (P3ID) as follows: A seller has k indistinguishable units of a good to sell. There are n customers that demand different quantities of the good. Each customer has a private valuation for the entirety of her demand, and the seller has a probabilistic model of this valuation. Formally the seller has the following inputs:

- $\boldsymbol{k}$: the number of indistinguishable goods available to sell.
- $\boldsymbol{n}$: the number of customers that have expressed demand for the good.
- $\boldsymbol{q_i}$: the number of units demanded by the $\boldsymbol{i}$th customer.
- $\boldsymbol{G_i(v_i)}$: a cumulative density function indicating the probability that the $\boldsymbol{i}$th customer draws a valuation below $\boldsymbol{v_i}$. Consequently, $\boldsymbol{1 - G_i(p)}$ is the probability that the customer will be willing to purchase her demand at price $\boldsymbol{p}$.

The seller wishes to make optimal discriminatory take-it-or-leave-it offers to all customers simultaneously. We make the following two assumptions as part of the P3ID to simplify the problem of choosing prices:

- **Continuous Probabilistic Demand (CPD) Assumption:** For markets involving a large number of customers, we can assume that the customer cumulative probability curves can be treated as continuous demand curves. In other words if a customer draws a valuation greater than or equal to \$1000 with probability $\frac{1}{2}$, we assume the customer demands $\frac{1}{2}$ of her actual demand at that price. This is formally modeled by the probabilistic demand of customer i at price p, $q_i * (1 - G_i(p))$.
- **Expected Supply (ESY) Assumption:** We assume that the seller maintains a strict policy against over-offering supply in expectation by limiting the number of goods sold to k (the supply). Note that k is not necessarily the entirety of the seller's inventory.

Under these assumptions, the goal of the seller is to choose a price to offer each customer, p_i, that maximizes the expected total revenue function, $F(p)$:

$$F(p) = \sum_i (1 - G_i(p_i)) * q_i * p_i \tag{1}$$

Subject to the ESY constraint that supply is not exceeded in expectation:

$$\sum_i (1 - G_i(p_i)) * q_i \leq k \tag{2}$$

3.2 P3ID and CKP Equivalence

To demonstrate the equivalence between the P3ID and CKP we will show that an instance of either can easily be reduced to an instance of the other. CKP instances involve a knapsack with a finite capacity, k, and a set of n items. Each item has a reward function, $f_i(x)$, and a weight w_i. Including a fraction x_i of item i in the knapsack yields a reward of $f_i(x_i)$ and consumes $w_i * x_i$ of the capacity.

We can easily reduce a P3ID instance to a CKP instance using the following conversion:

– Set the knapsack capacity to the seller's capacity in the P3ID instance.

$$k^{\mathtt{CKP}} = k^{\mathtt{P3ID}}$$

– Include one item in the CKP instance for each of the n customers in the P3ID instance.
– Set the weight of the ith item to the customer's demanded quantity in the P3ID instances.

$$w_i = q_i$$

– Set the reward function of the ith item to be the inverse of the seller's expected revenue from customer i.

$$f_i(x) = G_i^{-1}(1 - x) * x * q_i$$

The fraction of each item included in the optimal solution to this CKP instance, x_i^*, can be converted to an optimal price in the P3ID instance, p_i^*, using the inverse of the CDF function over customer valuations,

$$p_i^* = G_i^{-1}(1 - x_i^*)$$

To reduce a CKP instance to a P3ID instance we can reverse this reduction. The CDF function for the new P3ID instance is defined as,

$$G_i(p) = 1 - \frac{f_i^{-1}(p)}{p * q_i}$$

Once found, the optimal price for a customer, p_i^*, can be translated to the optimal fraction to include, x_i^*, using this CDF function,

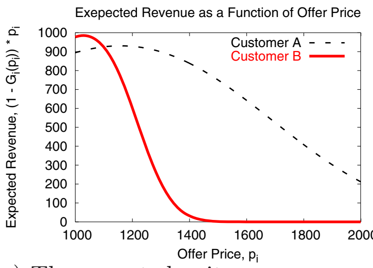
$$x_i^* = G_i(p_i^*)$$

This equivalence does not hold if either the CDF over customer valuations in the P3ID instance, or the reward function in the CKP instance is not invertible. However, if the inverse exists but is difficult to compute numerically, it can be approximated to arbitrary precision by precomputing a mapping from inputs to outputs.

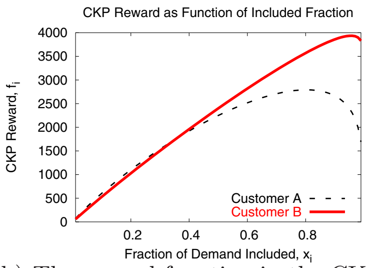
3.3 Example Problem

We provide this simple example to illustrate the kind of pricing problem we address in this paper, and its reduction to a CKP instance. Our example involves a PC Manufacturer with $k = 5$ finished PCs of the same type. Two customers have submitted requests for prices on different quantities of PCs[3] Customer A demands 3 PCs and Customer B demands 4 PCs. Each customer has a private valuation, if the manufacturer's offer price is less than or equal to this valuation the customer will purchase the PCs.

Based on public attributes that the Customers have revealed, the seller is able to determine that Customer A has a normal unit-valuation (price per unit) distribution with a mean of \$1500 and a standard deviation of \$300, $g_A = \mathcal{N}(1500, 300)$, and Customer B has a normal unit-valuation distribution with mean of \$1200, and a standard deviation of \$100, $g_B = \mathcal{N}(1200, 100)$. Figure 1(a) shows the expected revenue gained by the seller from each customer as a function of the offer price according to these valuation distributions. Figure 1(b) shows the reward functions for the corresponding CKP instance as a function of the fraction of the customer's demand included in the knapsack.



(a) The expected unit-revenue generated for the seller by each customer as a function of the customer's offer price $((1 - G_i(p_i))p_i))$, with p_i between 1000 and 2000.



(b) The reward function in the CKP instance corresponding to the expected revenue curves in Figure 1(a). Reward is presented as a function of the fraction of demand included in the knapsack.

Fig. 1. The expected customer revenue and corresponding reward for the example problem in Section 3.3

Note that in this example, as the price offered to Customer A (or Customer B) increases the probability (or Customer B) accepting it decreases, and hence so does the expected number of PCs sold to that customer. The manufacturer wishes to choose prices to offer each customer to maximize his overall expected revenue, and sell less than or equal to 5 PCs in expectation. In the following Section we will show how the optimal pricing solution to problems of this form can be computed.

[3] Although we have previously indicated that the CPD assumption made in our pricing formulation tends to hold only in large markets (i.e. more than 2 customers) our example is intentionally smaller for explanatory purposes.

In this example it turns out that the optimal solution is for the manufacturer to offer a unit price of \$1413 to Customer A, which has about a 58% chance of being accepted, and a price of \$1112 to Customer B which has about an 81% chance of being accepted. The total expected revenue of this solution is about \$1212 per unit and it sells exactly 5 units in expectation.

4 Asymmetric Concave CKPs

4.1 Characterizing an Optimal Solution

The main idea behind our algorithm for solving asymmetric CKPs is to add items to the knapsack according to the rate, or first derivative, of their reward functions. We will show that, if all reward functions are strictly concave,[4] and differentiable they share a unique first derivative value in an optimal solution. Finding the optimal solution amounts to searching for this first derivative value. To formalize and prove this proposition we introduce the following notations,

– Let $\boldsymbol{\phi_i(x) = f'_i(x)\frac{1}{w_i}}$, be the first derivative of the $\boldsymbol{i}$'th item's *unit* reward function. Item $\boldsymbol{i}$'s unit reward function is its reward per weight unit.
– Let $\boldsymbol{\phi_i^{-1}(\Delta)}$, be the inverse of the first derivative of $\boldsymbol{i}$'th item's unit reward function. In other words, it returns the fraction of the $\boldsymbol{i}$'th item where its unit reward is changing at the rate $\boldsymbol{\Delta}$.

Proposition 1. *Given a CKP instance, K, if all f_i in K are strictly concave and differentiable over the interval $[0,1]$, then there exists a unique Δ^* such that, $x_i^* = \phi_i^{-1}(\Delta^*)$, where x_i^* is the fraction of the i'th item in an optimal solution to K.*

Proof. First we will prove that $\phi_i(x)$ is invertible, and that $\phi_i^{-1}(\Delta^*)$ is unique for all i. The reward functions and unit reward functions (since these are simply scaled versions of the originals) in the CKP instance are strictly concave and differentiable on the interval $[0,1]$, by the predicate of our proposition. In other words, the first derivative of each unit reward function, $\phi_i(x)$, is decreasing and unique on the interval $[0,1]$. Because each unit reward function's first derivative is continuous, decreasing, and unique, it is invertible, and its inverse, $\phi_i^{-1}(\Delta)$, is unique.[5]

We will now prove that the unit reward functions of any two items, i and j, must share the same first derivative value in the optimal solution. To do this we introduce the following Lemma,

Lemma 1. *If f_i is strictly concave over the interval $[0,1]$, $\phi_i^{-1}(\Delta)$ increases as Δ decreases from $\phi_i(0)$ to 0.*

[4] Section 5.1 explains why we can reasonably restrict our consideration to concave reward functions in reductions from P3ID instances.

[5] This inverse may be difficult to characterize numerically. However, the precomputation technique suggested for approximating the inverse of G_i or f_i applies to ϕ_i as well.

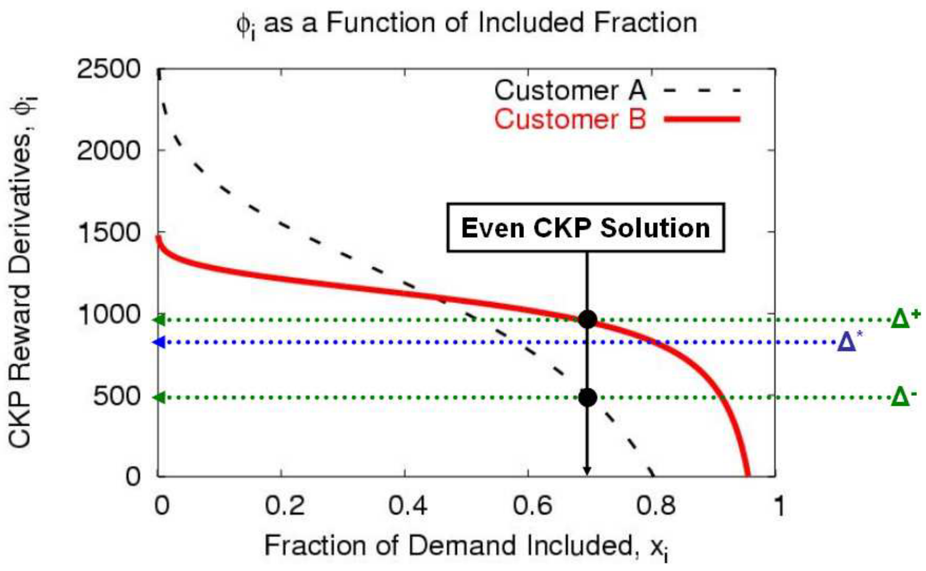


Fig. 2. Initial values for Δ^+ and Δ^- are computed from the `even_CKP` solution for the example problem in Section 3.3

Essentially the Lemma states that as the derivative of item i's unit reward function increases, the fraction of the item included in the knapsack shrinks. This is true because, as we have shown, the derivative is decreasing and unique.

For the remainder of the proof there are two cases we must consider:

Case 1: *the knapsack is not full in the optimal solution.* In this case the unit reward functions will all have derivatives of 0, since every item is included up to the point where its reward begins to decrease.[6]

Case 2: *the knapsack is full in the optimal solution.* In this case we will assume that f_i and f_j do not share the same derivative value, and show this assumption leads to a contradiction. Specifically, we can assume, *without loss of generality*, that the reward function of item i has a larger first derivative than j, i.e. $\phi_i(x_i^*) > \phi_j(x_j^*)$. Therefore, there must exist some ϵ, such that adding it to item j's unit reward derivative maintains the inequality, $\phi_i(x_i^*) > \phi_j(x_j^*) + \epsilon$. We can then construct an alternative solution to K as follows:

– Set x_j in our alternative solution to be the fraction of item j that provides its original derivative plus ϵ,

$$x_j' = \phi_j^{-1}(\phi_j(x_j^*) + \epsilon)$$

[6] We assume that all reward functions have derivatives ≤ 0 when an item is entirely included in the knapsack, since the item cannot possibly provide any additional reward.

– By Lemma 1 we know that $x'_j < x^*_j$, which provides some excess space, α, in the knapsack, $\alpha = w_j(x^*_j - x'_j)$. We can fill the empty space with item i, up to the point where the knapsack is full, or its derivative decreases by ϵ,

$$x'_i = \min\left(x^*_i + \frac{\alpha}{w_i}, \phi_i^{-1}(\phi_i(x^*_i) - \epsilon)\right)$$

It must be that $x'_i > x^*_i$. Either all of the knapsack space from item j was added, in which case the fraction of item i clearly increased. Otherwise, its derivative value decreased by ϵ, which, by Lemma 1, must have increased its included fraction. If $\phi_i(x'_i)$ decreased by ϵ before the knapsack filled up, we can reallocate the excess space to j,

$$x'_j = (k - x_i)\,\frac{1}{w_j}$$

Notice that we have constructed our alternate solution by moving the same number of knapsack units from item j to item i. In our construction we guaranteed that item i was gaining more reward per unit during the entire transfer. Therefore, the knapsack space is more valuable in the alternate solution. This contradicts our assumption that x^*_i and x^*_j were part of an optimal solution.

We have shown that any two unit reward functions must share the same derivative value, Δ^*, in an optimal solution. This implies that *all* unit reward functions must share the derivative value in an optimal solution (since no two can differ).

4.2 Finding Δ^*

In our proof of Proposition 1 we showed that $\Delta^* \geq 0$. We also showed that as Δ increases, the fraction of each item in the knapsack decreases. Thus, one method for finding Δ^* would be to begin with $\Delta = 0$ and increment by ϵ until the resulting solution is feasible (fits in the knapsack). However, much of this search effort can be reduced by employing a binary search technique.

Figure 3 presents pseudo-code for a binary search algorithm that finds solutions provably within ϵ of an optimal reward value. The algorithm recursively refines its upper and lower bounds on Δ^*, Δ^+ and Δ^-, until the reward difference between solutions defined by the bounds is less than or equal to ϵ.

The initial bounds, shown in Figure 2, are derived from a simple feasible solution where the same fraction of each item is included in the knapsack (see `even_CKP` in Figure 3). The largest derivative value in this solution provides the upper bound, Δ^+. This is because we can reduce the included fractions of each item to the point where all of their derivatives equal Δ^+, and guarantee the solution is still feasible. By the same reasoning, the smallest derivative value in the simple solution provides a lower bound Δ^-. Figure 2 shows how initial values of Δ^+ and Δ^- are computed from the even solution on the Example problem from Section 3.3.

```
procedure $\epsilon$-opt_CKP($K$)
    $x \leftarrow$ even_CKP($K$)
    $\Delta^+ \leftarrow \max_i \phi_i^{-1}(x_i)$
    $\Delta^- \leftarrow \min_i \phi_i^{-1}(x_i)$
    return binary_search($\Delta^+, \Delta^-, K$)

procedure binary_search($\Delta^+, \Delta^-, K$)
    if converged($\Delta^+, \Delta^-, K$) then
        $x^+ \leftarrow \{\phi_1^{-1}(\Delta^+), \ldots, \phi_n^{-1}(\Delta^+)\}$
        return $x^+$
    end if
    $\delta \leftarrow \frac{\Delta^+ - \Delta^-}{2}$
    if feasible($\{\phi_1^{-1}(\delta), \ldots, \phi_n^{-1}(\delta)\}$, $K$) then
        return binary_search($\delta, \Delta^-, K$)
    else
        return binary_search($\Delta^+, \delta, K$)
    end if

procedure even_CKP($K$)
    $\hat{w} \leftarrow \sum_i w_i$
    return $\{\frac{k}{\hat{w}}, \ldots, \frac{k}{\hat{w}}\}$

procedure feasible($x, K$)
    return $\sum_i w_i x_i \le k$

procedure converged($\Delta^+, \Delta^-, K$)
    $x^+ \leftarrow \{\phi_1^{-1}(\Delta^+), \ldots, \phi_n^{-1}(\Delta^+)\}$
    $x^- \leftarrow \{\phi_1^{-1}(\Delta^-), \ldots, \phi_n^{-1}(\Delta^-)\}$
    return $\sum_i f_i(x^+) - f_i(x^-) \le \epsilon$
```

Fig. 3. Pseudo-code for an ϵ-optimal concave CKP binary search algorithm

During each iteration, a new candidate bound, δ, is computed by halving the space between the prior bounds. The process continues recursively: if the new bound defines a feasible solution it replaces the old upper bound, otherwise (if it is not a valid upper bound), it replaces the old *lower* bound.

When the algorithm converges the solution defined by Δ^+ is guaranteed to be feasible and within ϵ of the optimal solution. Convergence is guaranteed since we have proved that Δ^* exists, and the bounds get tighter after each iteration. It is difficult to provide theoretical guarantees about the number of iterations, since convergence is defined in terms of the instance-specific reward functions. However, the empirical results in Section 6 show that the algorithm typically converges exponentially fast in the number of feasibility checks.

4.3 Shared Resource Extension

Our ϵ-optimal binary search algorithm can be extended to solve problems involving more complex resource constraints than typically associated with CKPs. In particular, the algorithm can be generalized to solve reductions of Probabilistic Pricing Problems with Shared Resources (P3SR). P3SR instances involve sellers with multiple *distinguishable* goods for sale. Each good in a P3SR consumes some amount of finite shared resources, such as components or assembly time. This model allows for techniques capable of supporting the movement from make-to-stock practices to assemble-to-order or make-to-order practices.

By applying the reduction described in Section 3.2, a P3SR instance can be converted to a problem similar to a CKP instance. However, the resource constraint in the resulting problem is more complex than ensuring that a knapsack contains less than its weight limit. It could involve determining the feasibility of a potentially $\mathcal{NP}$-Hard scheduling problem, in the case of a shared assembly line and customer demands with deadlines. Clearly, this would require, among other things, changing the feasibility checking procedure (see `feasible()` in Figure 3), and could make each check substantially more expensive.

5 Customer Valuations

5.1 Diminishing Returns Property

Our algorithm was designed to solve CKP reductions of P3ID instances. Recall that it applies only when the reward functions are concave over the interval $[0, 1]$. This is not a particularly restrictive requirement. In fact, this is what economists typically refer to as the Diminishing Returns[7] (DMR) property. This property is generally accepted as characterizing many real-world economic processes [2].

Definition: *The DMR property is satisfied for a P3ID instance when, for a given increase in any customer's filled demand, the increase in the seller's expected revenue is less per unit than it was for any previous increase in satisfaction that customer's demand.*

Note that our market model also captures the setting where customer valuations are determined by bids from competing sellers. In this setting normally distributed competing bid prices can also be shown to result in concave reward functions. This situation is representative of environments where market transparency leads sellers to submit bids that hover around a common price.

5.2 Normal Distribution Trees

We consider a technique which a seller may use to model a customer's valuation distribution. It will use a normal distribution to ensure our model satisfies the desired DMR property. We assume that customers have some public attributes, and the seller has historical data associating attributes vectors with valuations.

Our technique trains a regression tree to predict a customer's valuation from the historical pricing data. A regression tree splits attributes at internal nodes, and builds a linear regression that best fits the training data at each leaf. When a valuation distribution for a new customer needs to be created, the customer is associated with a leaf node by traversing the tree according to her attributes. The prediction from the linear model at the leaf node is used as the mean of a normal valuation distribution, and the standard deviation of the distribution is taken from training data that generated the leaf.

[7] This is also occasionally referred to as the Decreasing Marginal Returns property.

Formally the regression tree learning algorithm receives as input,

– $\boldsymbol{n}$: the number of training examples.
– $\boldsymbol{a_i}$: the attribute vector of the $\boldsymbol{i}$'th training example.
– $\boldsymbol{v_i}$: the valuation associated with the $\boldsymbol{i}$'th training example.

A regression tree learning algorithm, such as the M5 algorithm [10], can be used to learn a tree, T, from the training examples. After the construction of T, the j'th leaf of the tree contains a linear regression over attributes, $y_j(a)$. The regression is constructed to best fit the training data associated with the leaf. The leaf also contains the average error over this data, s_j.

The regression tree, T, is converted to a distribution tree by replacing the regression at each node with a normal distribution. The mean of the normal distribution at the j'th leaf is set to the prediction of the regression, $\mu_j = y_j(a)$. The standard deviation of the distribution at the j'th leaf is set to the average error over training examples at the leaf, $\sigma_j = s_j$. Figure 4 shows an example of this kind of normal distribution tree.

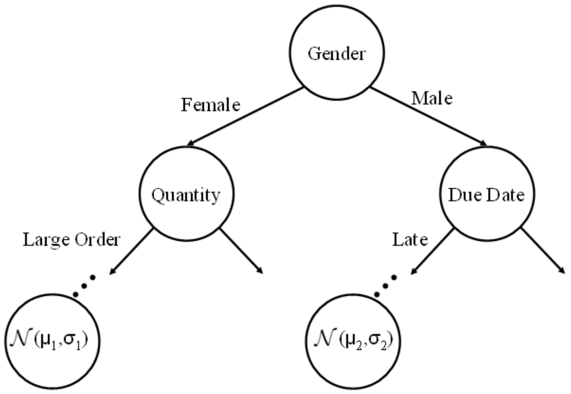


Fig. 4. An example Normal Distribution Tree

5.3 Learning Customer Valuations in TAC

TAC SCM provides an ideal setting to evaluate the distribution tree technique described in the previous section. Each customer request in TAC SCM can be associated with several attributes. The attributes include characterizations of the request, such as its due date, PC type, and quantity. The attributes also include high and low selling prices for the requested PC type from previous simulation days. Upon the completion of a game, the price at which each customer request was filled is made available to agents. This data can be used with the technique described in the previous section to train a normal distribution tree. The tree can then be used in subsequent games to construct valuation distributions from request attributes.

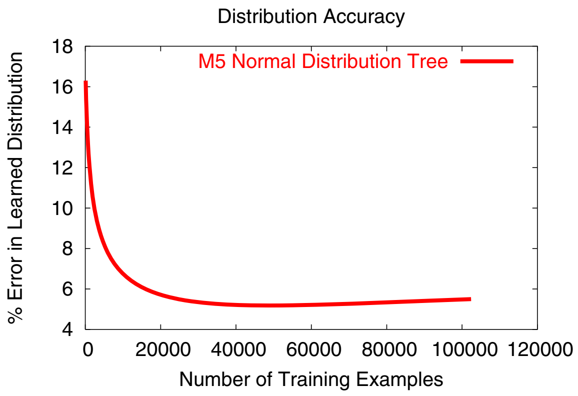


Fig. 5. The accuracy curve of an M5 normal distribution tree as the number of training instances increases

Figure 5.3 shows the accuracy curve of a normal distribution tree trained on historical data with an M5 learning algorithm. Training instances were drawn randomly from customer requests in the 2005 Semi-Final round of TAC SCM and testing instances were drawn from the Finals. The attributes selected to characterize each request included: the due date, PC type, quantity, reserve price, penalty, day on which the request was placed, and the high and low selling prices of the requested PC type from the previous 5 game days.

The error of the distribution was measured in the following way: starting at $p = .1$, and increasing to $p = .9$, the trained distribution was asked to supply a price for all test instances that would fall below the actual closing price (be a winning bid) with probability p. The average absolute difference between p and the actual percentage of test instances won was considered the error of the distribution. The experiments were repeated with 10 different training and testing sets. The results show that normal distribution trees can be used to predict distributions over customer valuations in TAC SCM with about 95%, accuracy after about 25,000 training examples.

6 Empirical Evaluation

6.1 Empirical Setup

Our experiments were designed to investigate the convergence rate of the ϵ-optimal binary search algorithm. We generated 100 CKP instances from P3ID instances based on the pricing problem faced by agents in TAC SCM. The P3ID instances were generated by randomly selecting customer requests from the final round of the 2005 TAC SCM. Each customer request in TAC SCM has a quantity randomly chosen uniformly between 1 and 20 units. Normal probability distributions were generated to approximate the customer valuations of each customer using the technique described in Section 5 with an M5 Regression Tree

```
procedure greedy_CKP($K$)
  converged $\leftarrow \perp$
  while $\neg$converged and $\sum_i x_i < n$ do
    $i^* \leftarrow \mathrm{argmax}_i$ unit_reward_increase $(i, x_i, K)$
    $\delta^* \leftarrow$ best_increase($i^*, x_{i^*}, K$)
    if feasible($x_{i^*} + \delta^*, K$) then
      $x_{i^*} \leftarrow x_{i^*} + \delta^*$
    else
      $x_{i^*} \leftarrow x_{i^*} + \frac{1}{w_i}\left(k - \sum_i x_i w_i\right)$
      converged $\leftarrow \top$
    end if
  end while
  return $\{x_1, \ldots, x_n\}$

procedure unit_reward_increase($i, x_i, \delta, K$)
  $\delta^* \leftarrow$ best_increase($i, x_i, K$)
  return $\frac{1}{w_i \delta^*}\left(f_i(x_i + \delta^*) - f_i(x_i)\right)$

procedure best_increase($i, x_i, K$)
  return $\mathrm{argmax}_\delta \frac{f_i(x_i+\delta) - f_i(x_i)}{\delta}$
```

Fig. 6. Pseudo-code for the greedy heuristic algorithm used by the 2005 first placed agent, TacTex

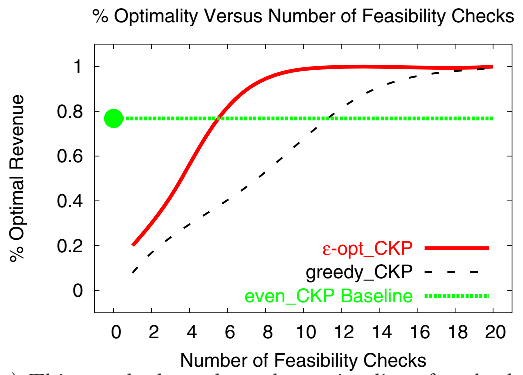
learning algorithm. The learning algorithm was given 50,000 training instances from the 2005 TAC SCM Semi-Final rounds.

We tested our algorithm against the even solution, which allocates equal resources to each customer, and the greedy heuristic algorithm used by the first place agent, TacTex [9]. Figure 6.1 provides pseudo-code adapting the TacTex algorithm to solve the P3ID reductions. It greedily adds fractions of items to the knapsack that result in the largest increases in expected unit-revenue.

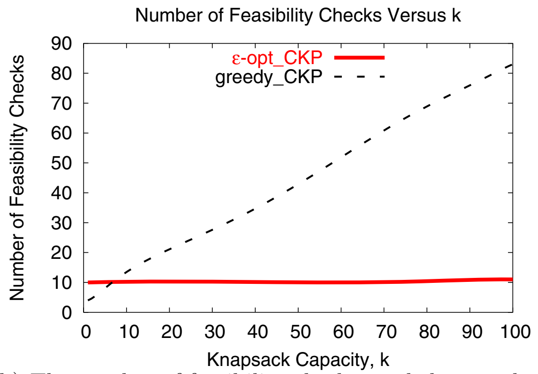
We performed three sets of experiments. The first set of experiments provided each algorithm with 20 PCs to sell in expectation, and the same 200 customer requests (this represents a pricing instance of a TAC SCM agent operating under a make-to-stock policy). Figure 7(a) shows each algorithm's percentage of an optimal expected revenue after each feasibility check. For the second set of experiments, the algorithms were given 200 customer requests, and their PC supply was varied by 10 from $k = 10$, to $k = 100$. Figure 7(b) shows the number of feasibility checks needed by the binary search and greedy algorithms to reach solutions within 1% of optimal. The last set of experiments fixed $k = 20$ and varied n by 100 from $n = 200$ to $n = 1000$. Figure 7(c) shows the number of feasibility checks needed by each algorithm to reach a solution within 1% of optimal as n increased.

6.2 Empirical Results

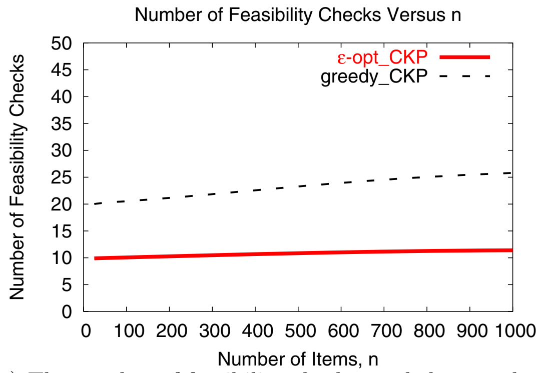
The results presented in Figure 7 compare the optimality of the CKP algorithms to the number of feasibility checks performed. This comparison is important to investigate for two reasons, i.) because it captures the convergence rate of the algorithms, and ii.) because these algorithms are designed to be extended to



(a) This graph shows how the optimality of each algorithm improves with each feasibility check it uses.



(b) The number of feasibility checks needed to reach a solution within 1% of optimal as k increases.



(c) The number of feasibility checks needed to reach a solution withing 1% of optimal as n increases.

Fig. 7. Performance of CKP algorithms on instances reduced from TAC SCM pricing problems. Unless otherwise specified, results are averaged over 100 CKP instances with $n = 200$ and $k = 20$.

shared resource settings discussed in Section 4.3 where each feasibility check involves solving (or approximating) an $\mathcal{NP}$-Hard scheduling problem.

The first set of results, shown in Figure 7(a), confirms that the ϵ-optimal binary search algorithm converges exponentially fast in the number of consistency checks. In addition, the results confirm the intuition of Pardoe and Stone in [9] that the greedy heuristic finds near optimal solutions on CKP instances generated from TAC SCM. However, the results also show that it has a linear, rather than exponential, convergence rate in terms of consistency checks. This indicates that our binary search algorithm scales much better than the greedy technique. Finally, the first set of results shows that the even solution, which does not use consistency checks, provides solutions to TAC SCM instances that are about 80% optimal on average.

Figures 7(b) and 7(c) investigate how the number of feasibility checks needed to find near (within 99% of) optimal solutions changes as the supply and number of customers increase. The even solution is not included in these results because it does not produce near optimal solutions. The results shown in Figure 7(b) show that the number of consistency checks used by the greedy algorithm increases linearly with the size of the knapsack, whereas the convergence rate of the binary search algorithm does not change. The results shown in Figure 7(c) show that the number of consistency checks used by both algorithms does not significantly increase with the number of customers.

7 Conclusion

In this paper we presented a model for the problems faced by sellers that have multiples copies of an indistinguishable good to sell to multiple customers. We have modeled this problem as a Probabilistic Pricing Problem with Indistinguishable Goods (P3ID) and formally shown its equivalence the Continuous Knapsack Problem (CKP). We showed that P3ID instances with customer valuation distributions that satisfy the DMR property reduce to CKP instances with arbitrary concave reward functions. Prior work had not addressed CKP instances with asymmetric *nonlinear* concave reward functions. To address this gap, we provided a new ϵ-optimal algorithm for such CKP instances. We showed that this algorithm converges exponentially fast in practice. We also provide a technique for learning normal distributions of customer valuations from historical data, by extending existing regression tree learning algorithms. We validated our distribution learning technique and our binary search technique for the P3ID on data from 2005 TAC SCM. Our results showed that our learning technique achieves about 95% accuracy in this setting, indicating that TAC SCM is a good environment in which to apply our P3ID model. Our results further showed that our binary search algorithm for the P3ID scales substantially better than a technique adapted from the winner of the 2005 TAC SCM competition.

Acknowledgments

The research reported in this paper has been funded by the National Science Foundation under ITR Grant NSF-0205435 and IGERT Grant NSF-DGE99762.

References

1. M. Benisch, A. Greenwald, I. Grypari, R. Lederman, V. Naroditskiy, and M. C. Tschantz. Botticelli: A supply chain management agent. In *Proceedings of AAMAS '04*, pages 1174–1181, New York, July 2004.
2. K. E. Case and R. C. Fair. *Principles of Economics (5th ed.)*. Prentice-Hall, 1999.
3. J. Collins, R. Arunachalam, N. Sadeh, J. Eriksson, N. Finne, and S. Janson. The supply chain management game for the 2005 trading agent competition. Technical Report CMU-ISRI-04-139, Carnegie Mellon University, 2005.
4. O. Etzioni, R. Tuchinda, C. A. Knoblock, and A. Yates. To buy or not to buy: mining airfare data to minimize ticket purchase price. In *Proceedings of KDD'03*, pages 119–128, New York, NY, USA, 2003. ACM Press.
5. R. Ghani. Price prediction and insurance for online auctions. In *Proceedigns of KDD'05*, pages 411–418, New York, NY, USA, 2005. ACM Press.
6. H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems*. Springer, 2004.
7. D. Lawrence. A machine-learning approach to optimal bid pricing. In *Proceedings of INFORMS'03*, 2003.
8. A. Melman and G. Rabinowitz. An efficient method for a class of continuous knapsack problems. *Society for Industrial and Applied Mathematics Review*, 42(3): 440–448, 2000.
9. D. Pardoe and P. Stone. Bidding for customer orders in TAC SCM. In *Proceedings of AAMAS-04 Workshop on Agent-Mediated Electronic Commerce*, 2004.
10. J. R. Quinlan. Learning with Continuous Classes. In *5th Australian Joint Conference on Artificial Intelligence*, pages 343–348, 1992.
11. A. G. Robinson, N. Jiang, and C. S. Lerme. On the continuous quadratic knapsack problem. *Math. Program.*, 55(1-6):99–108, 1992.
12. T. Sandholm and S. Suri. Market clearability. In *Proceedings of IJCAI'01*, pages 1145–1151, 2001.

Agents' Bidding Strategies in a Combinatorial Auction Controlled Grid Environment

Michael Schwind[1], Tim Stockheim[2], and Oleg Gujo[2]

[1] Business Information Systems and Operations Research, Technical University Kaiserslautern, Gottlieb-Daimler-Strasse 47, D-67663 Kaiserslautern, Germany
schwind@wiwi.uni-kl.de

[2] Institute of Information Systems, Johann Wolfgang Goethe University, Mertonstrasse 17, D-60054 Frankfurt, Germany
{stockheim,gujo}@is-frankfurt.de

Abstract. In this article we present an *agent-based simulation environment* for task scheduling in a grid-like computer system. The scheduler allows to one *simultaneously allocate resources* such as CPU time, communication bandwidth, volatile and non-volatile memory by employing a combinatorial resource allocation mechanism. The allocation is performed by an *iterative combinatorial auction* in which *proxy-bidding agents* try to acquire their desired resource allocation profiles with respect to limited monetary budget endowments. To achieve an efficient allocation process, the auctioneer provides *resource price information* to the bidders. We use a pricing mechanism based on shadow prices in a *closed loop* system in which the agents use monetary units awarded for the resources they provide to the system for the acquisition of complementary capacity. Our objective is to identify optimal bidding strategies in the multi-agent setting with respect to varying preferences in terms of resource quantity and waiting time for the resources. Based on a utility function we characterize two types of agents: a *quantity maximizing agent* with a low preference for fast bid acceptance and an *impatient bidding agent* with a high valuation of fast access to the resources. By evaluating different strategies with varying initial bid pricing and price increments, it turns out that for quantity maximizing agents patience and low initial bids pay off, whereas impatient agents should avoid high initial bid prices.

1 Introduction

The agent-based simulation environment for resource allocation in grid-like systems presented here employs a combinatorial task scheduler that enables the simultaneous allocation of resources like CPU time, communication bandwidth, volatile and non-volatile memory. In contrast to traditional grid allocation approaches, our allocation process considers production complementarities and substitutionalities for these resources making the resulting resource usage much more efficient. The central scheduling instance of our system is comparable to an auctioneer that performs an iterative combinatorial auction in which proxy-agents try to acquire the resources needed in computational tasks for the provision of *information services and information production (ISIP)* by submitting package bids for the resource combinations. The proxy-agents' *willingness-to-pay (W2P)* for these bundles is constrained by limited budgets of the virtual currency they are endowed with. The allocation system simulates a *closed loop* grid economy in which the agents gain monetary units for resources they provide to other grid system participants via the auctioneer. The virtual currency earned can be used for the acquisition of complementary resource capacity. The simulation environment allows the utilization and benchmarking of different proxy-bidding strategies in various situations. Two main bidding strategies are compared here:

– An *impatient bidding agent* that tries to achieve a quick acceptance of bids by using a fast inclining bid price in the subsequent rounds.
– A *quantity maximizing agent* that submits bid bundles at low initial prices and waits for bid acceptance, while slowly increasing the price in the following round in case of bid rejection.

The bidding strategies are compared under changing resource availability situations with respect to the resulting allocation quality that is measured in terms of received utility, defined by a utility function that allows one to represent the different preferences of the agents.

2 Combinatorial Auctions in Grid Environments

The use of the computing power provided by distributed computer infrastructure, such as the grid, is of increasing interest for *information technology* infrastructure and service providers [1]. The simultaneous use of network resources and computing capacity to enable *web-based video conference* and *peer-to-peer telecommunication services* between corporations is an example of such a *business-to-business*

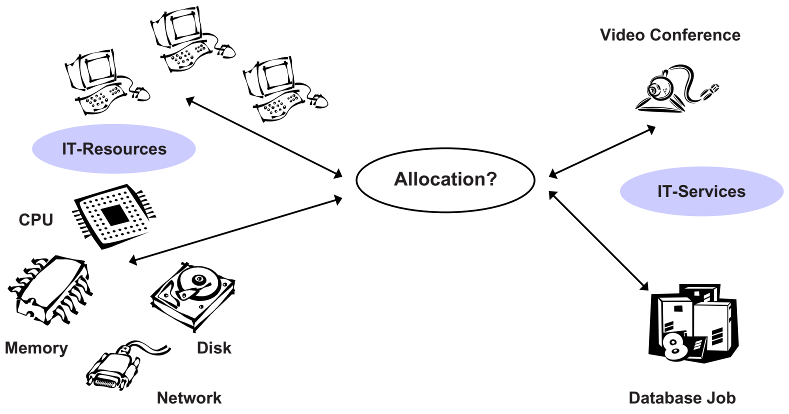


Fig. 1. Scenario for the allocation of resources in a distributed IT infrastructure

related application of ISIP tasks [2]. An exemplary scenario for the allocation of IT services to distributed IT resources is depicted in figure 1.

Requests for the resources that are necessary to provide services like a video conference or database processing job have to be allocated to various networked PC systems including different resource types. Searching for usable economically inspired mechanisms for the *price controlled resource allocation (PCRA)* auctions seem to be a good candidate [3]. While using this mechanism the preferences of the users and their W2P play an important role: whereas the data processing service might not be very time critical for the user, the video conference is. Accordingly, the video conference users should be willing to pay more for the immediate availability of the resources than the user of the database service. The transfer of a considerable amount of data that is collected during daily business activities, in times of low infrastructure resource load, could be seen as a further instance of an IT task that has no high time criticality and where a PCRA is valuable for the users and the service providers [4,5].

CA are a suitable tool for allocating interdependent resources according to the W2P of the participants. The ISIP process of our example in figure 1 comprises an allocation problem with strong complementarities. For example, if the video conference service via the web in our scenario is processed on different computers and acquires CPU time without obtaining communication network capacity between these computers at the same time, the acquired CPU time is worthless. The application of CAs for resource allocation in distributed computer systems is still in its infancy despite its excellent applicability to grid computing. Recent approaches make use of a periodically performed CA with very simple allocation mechanisms to achieve sufficient real time performance [6,7]. The use of a budget of a virtual currency available for the bidders for task procurement purposes is often used to constrain the liquidity in these allocation systems [8]. Neumann et al. [9] include quality attributes in the combinatorial allocation process for a grid system which is close to our approach. However, most proposed systems either suffer from performance problems or have to accept severe constraints for the formulation of resource bundles in the grid.

3 An Agent-Based Simulation Environment for Combinatorial Resource Allocation in Grid Systems

The combination of grid and multi-agent technology seems to be a viable approach in the application context described above [10]. Our combinatorial grid scheduling environment is therefore based on the agent workbench JADE 3.3. It goes beyond the recent research approaches in several points:

– The system allows the use of several winner determination algorithms such as greedy, simulated annealing, genetic programming, and integer programming methods according to the users' requirements in terms of allocation quality and computation time [11].
– The simulator provides tools to investigate various bidding behaviors on the part of the proxy-agents. We will concentrate on this aspect in this paper.

– The framework can simulate changing resource capacities to test the grid scheduler's reaction with respect to allocation efficiency and system stability.

3.1 Scenario for a PCRA in a Combinatorial Grid System

This section gives a brief overview of the *resource allocation scenario* for ISIP provision used in our work. The scenario includes four resource types:

– *Central processing units (CPU)* that are mainly responsible for the processing of the data in the ISIP processes.
– *Volatile memory capacity (MEM)* which is necessary to store short-term processing data for the central processing units.
– *Non-volatile storage capacity (DSK)* which is necessary to keep mass data and to provide program codes for the execution of the ISIP processes.
– *Network bandwidth (NET)* that is required for the data interchange between different computer units.[1]

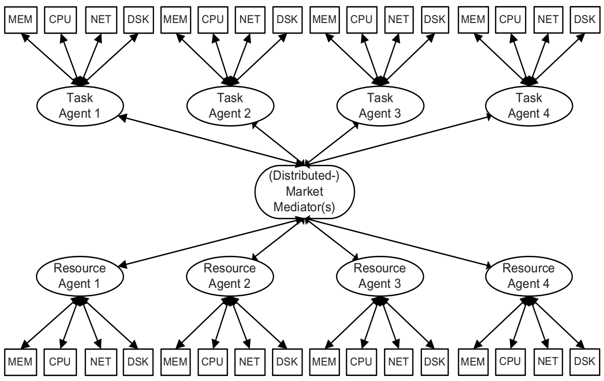


Fig. 2. Scenario for the allocation of ISIP resources

The scenario for our combinatorial grid simulator is constructed as follows:

– *Task agents* are engaged in acquiring the resources needed to process the ISIP task in the distributed computer system on behalf of real world clients. They do this by bidding for the required resource combination via the auctioneer.
– *A mediating agent* (auctioneer) receives the resource bids and calculates an allocation profile for the available resources managed by the *resource agents* according to the allocation mechanism. After a successful auction, process bidders are informed about the acceptance of their bids.

[1] Considering the fact that network connections themselves exhibit complementarities, connections should be treated as additional resources, one for each connection pair.

– *Resource agents* collect information about available resources on their particular host IT systems through a network of distributed computers and provide this information to the market mediator. The resource agents offer the available capacities to the task agents via the mediating agent. If a bid is accepted via the auctioneer, the resources thus acquired are reserved for the corresponding winning agent in advance.

Figure 2 depicts the ISIP allocation scenario. Resource agents administer available MEM, CPU, NET, and DSK capacities on their particular host computer systems on the supply side. On the demand side task agents collect the required resource combinations, including MEM, CPU, NET, and DSK capacity, needed to accomplish their production tasks. Between resource and task agents there is a market mediator that allocates the resources employing a combinatorial auction. For the formal representation of the bids, a two-dimensional *bid-matrix (BM)* is used. One dimension of the BM describes the time $t \in \{1, \ldots, T\}$ at which the resource is required within the request period. The other dimension $r \in \{1, \ldots, R\}$ denotes the resource types MEM, CPU, NET, DSK. The request for a quantity of an individual resource r at time t is then denoted by a matrix element $q_{i,j}(r, t)$. A price $p_{i,j}$ is assigned to each BM expressing the agent's W2P for the resource bundle. Indices i and j identify a specific bid matrix.

Table 1. Example of a bid matrix BM submitted by a task agent

	Time Slot t							
resource	1	2	3	4	5	6	7	8
1	2	2			3	3	3	3
2		1	1	1				
3		2	2		1	1	1	1
4	3	3				2	2	

The value q^{bmax} denotes the maximum resource load that can be requested by a bidder for a single BM element $q_{i,j}(r, t)$. These elements are supplied with *time slot occupation probability* p^{tso}. In addition to the BM, two other matrices play a role within our grid simulation framework:

– The *allocation matrix* (AM) describes the currently awarded allocation $q(r, t)$ for resources r and time slots t within the following ISIP provision period T.
– The *constraint matrix* (CM) expresses the maximum quantity $q^{max}(r, t)$ of resource r the auctioneer can assign to the task agents at time t. The maximum possible resource load of the CM represents the aggregated resource availability for the following time slots.

3.2 The Combinatorial Scheduling Auction

Following the description of the scenario, figure 3 illustrates the course of action of the system. The AUML sequence diagram depicts the message flow based on the FIPA definition of the English auction protocol.[2]

[2] www.fipa.org/specifications/fipa00031D

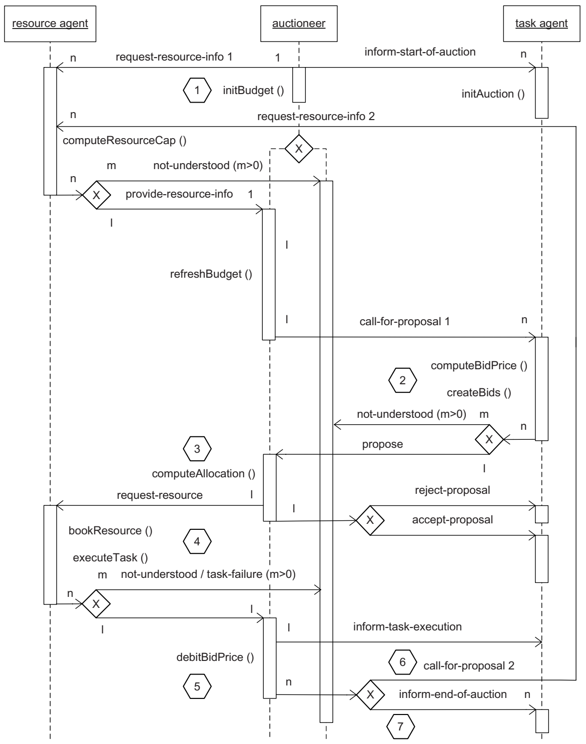


Fig. 3. FIPA AUML diagram for the iterative combinatorial scheduling auction

While in our closed economy each bidder has two roles (as provider and user of resources), figure 3 separately depicts both roles (*resource agent* and *task agent*) to provide a greater generalization and better readability. Each step is marked by a ⬡-symbol and detailed in the corresponding paragraph:

1. The auctioneer requests the resource agents to evaluate the available resource capacities and informs the bidders about the bidding terms. Then he awards an initial budget to the task agents and announces the start of the auction.
2. Following the auctioneer's call for proposals, the task agents create their bids according to the desired resource combination. Bidders compute the associated bid price, dependent on their actual pricing policy, their budget level, and the latest resource prices, if applicable.
3. The auctioneer receives the bids and calculates the return-maximizing combinatorial allocation. He informs the task agents about bid acceptance/ rejection and requests the resource agents to reserve the resources awarded.
4. Resource agents inform the auctioneer about the status of the task execution.
5. The auctioneer distributes task status information to the task agents and debits the bid price for the awarded bids from their accounts, followed by a call for proposals for the next round.

6. Task agents can renew their bids in the next round in case of non-acceptance or non-execution. The agents' bid pricing rules are defined in paragraph 3.5.
7. The process is repeated until the auctioneer announces the end of auction.

In the following we describe the elements of the combinatorial grid scheduler: the budget management, the auctioneer and the task agents' bidding behavior.

3.3 The System's Budget Management Mechanism

Each task agent holds a monetary budget that is initialized with a fixed amount BG^{ini} of *monetary units (MUs)* when the negotiations begin. At the beginning of each round k, the task agents' budgets are refreshed (see figure 3 - ①) with an amount of MUs enabling the agents to acquire the resource bundles required for their ISIP provision task. In order to avoid the expiry of the agents' budgets during the iterative auctioning process, the agents are integrated into a monetary circuit in the *closed-loop* grid economy regarded in this work. Task and resource agents act as a unit of consumer and producer both owning the resources of their peer system. This means a task and a resource agent reside simultaneously on each computer in the grid. The resource agent does the reporting of resource usage and provisioning for the task agent owning the peer computer resources (see figure 3 - ①, and ④). The agents on the peer computer are compensated for the resources provided to the system. Starting with the initial budgets BG^{ini} the amount of MUs circulating in the system is kept constant for the *closed grid economy*. The accounting of the agents' budgets in the grid system is done by the combinatorial auctioneer (see figure 3 - ①, and ⑤).

3.4 The Combinatorial Auctioneer

The combinatorial auctioneer controls the iterative allocation process of the grid system. For this purpose the auctioneer awaits the bids that have been submitted by the task agents for the current round. The bids that are submitted in the form of j XOR-bundled BMs in bid i are shown in table 1 and represent the task agents' requested capacity $q_{i,j}(r,t)$ of the resources r at a particular point of time t. After having received all alternative BMs submitted by the task agents, the auctioneer has to solve the *combinatorial auction problem (CAP)* which is NP-hard [12,13]. The CAP is often denoted as the *winner determination problem (WDP)*, according to the traditional auctioneer's task of identifying the winner. While the number of resources is $R \in \mathbb{N}$, the number of time slots is $T \in \mathbb{N}$, the number of bid bundles is $I \in \mathbb{N}$ and the number of XOR-bids in the bundles is $J_i \in \mathbb{N}$, the formal description of the CAP could be considered as a special variant of the *weighted set packing problem (WSPP)* [14] and is formulated as:

$$\begin{aligned} &\max \sum\nolimits_{i=1}^{I} \sum\nolimits_{j=1}^{J_i} p_{i,j}\ x_{i,j} \\ \text{s. t.}\quad & q(r,t) = \sum\nolimits_{i=1}^{I} \sum\nolimits_{j=1}^{J_i} q_{i,j}(r,t)\ x_{i,j} \leq q^{max}(r,t), \\ & \forall_{r\in\{1,\ldots,R\},t\in\{1,\ldots,T\}} \text{ and} \\ & \sum\nolimits_{j=1}^{J_i} x_{i,j} \leq 1, \quad \forall_{i\in\{1,\ldots,I\}}. \end{aligned} \tag{1}$$

Price of XOR-bundle j of bid i:	$p_{i,j}$	$\in \mathbb{R}^+$
Acceptance variable:	$x_{i,j}$	$\in \{0; 1\}$
Resource requests:	$q_{i,j}(r,t)$	$\in \mathbb{N}$

The auctioneer's primary goal is to maximize the income received under the limitation of the available resources and a maximum of one accepted bid per agent (equation 1). In order to accelerate the price-finding process, the auctioneer provides feedback on resource availability to the bidders to adjust their W2Ps $p_{i,j}$ for bids that have not been accepted in the previous rounds. As mentioned above, it is not always possible to calculate *unambiguous prices* (anonymous prices) for the individual resources in a combinatorial auction. In many cases, explicit resource prices can only be calculated for each individual bid. Kwasnica et al. [15] describe a pricing scheme for all individual goods in a combinatorial auction by approximating the prices in a divisible case based on an LP approach first proposed by Rassenti et al. [16]. As in a similar approach by Bjorndal and Jornsten [17] they employ the *dual solution* of the relaxed WDP to calculate the shadow prices. In the simulation model presented here, the dual LP approach of Kwasnica et al.[15] is adopted:[3]

$$\min\ z = \sum\nolimits_{r=1}^{R} \sum\nolimits_{t=1}^{T} q^{max}(r,t) \cdot sp_{r,t} \tag{2}$$

$$s.t. \quad \sum_{r=1}^{R} \sum_{t=1}^{T} q_{i,j}(r,t) \cdot sp_{r,t} + (1 - x_{i,j}) \cdot \delta_{i,j} = p_{i,j} \tag{3}$$

Reduced cost:	$\delta_{i,j}$	$\in \mathbb{R}_0^+$
Shadow price of one element:	$sp_{r,t}$	$\in \mathbb{R}_0^+$

The proposed SP calculation uses the *primal solution* for the LP problem delivered from open source LP solver *LPSOLVE 5.5*[4] for the determination of accepted bids [18]. As described above, each resource r has T available time slots per round. The SPs are weighted by the resource usage and grouped to

[3] The result of the following formula is denoted as reduced SPs. Omitting the rejected bids in the calculation of dual prices yields a higher result [17].

[4] http://www.geocities.com/lpsolve/

shadow prices for each resource. Now the *market value of a resource unit* can be calculated while using the shadow prices and summarizing the utilized capacity of each resource r for all accepted bids:

$$v_r = \frac{\sum_{t=1}^{T} sp_{r,t} \cdot q(r,t)}{\sum_{t=1}^{T} q(r,t)} \quad \forall_{r \in \{1,\dots,R\}} \tag{4}$$

Market value of a resource unit: $v_r \in \mathbb{R}_0^+$

In general bid prices are not assumed to be linear in this framework. This means that shadow prices $sp_{r,t}$ cannot be calculated by the auctioneer for each round, i.e. there is no solution of the LP problem [17]. In such cases the auctioneer relies on an approximation of the market values based on historic data $H_{sp}(r)$ which contains the market values calculated in the last n rounds:

$$\hat{v}_r = \frac{\sum_{h_{sp} \in H_{sp}(r)} h_{sp}}{|H_{sp}(r)|} \quad \forall_{r \in \{1,\dots,R\}} \tag{5}$$

Approximated market value: $\hat{v}_r \in \mathbb{R}_0^+$

Unfortunately, the shadow price calculation is computationally expensive. If the variation of required resources in the bid or the number of bids increase, the required time may exclude shadow price approximation as an alternative.

3.5 The Task Agents' Bidding Model

Based on the market values of resources v_r the task agents in the combinatorial simulation model try to acquire the resources needed for ISIP provision. Besides the market values of resources, the task agents' bidding behavior is determined by their budget and by an associated *bidding strategy*. The major goal of the task agents is to receive as many resource units as possible that are required for the performance of their ISIP processes at the lowest possible amount of MUs. The general bidding behavior of the task agents is similar for both pricing information methods, scarcity and shadow prices as well:

- In each round k the task agents generate M new bids. The task agents submit several bids as exclusively eligible bundles (OR-of-XOR). If the budget of a task agent is exhausted due to the continued acceptance of bids by the auctioneer, no further bids are submitted until the budget has recovered.
- The task agents repeat bidding for rejected bids in the following round while modifying the W2P with respect to the current pricing information.

Depending on the market value v_r of the resources required for the ISIP provision process, task agents have to formulate their W2P for the bids. To calculate the bundle prices, two cases must be considered:

$$p_{i,j}(k) = \begin{cases} BG^{ini} / (L \cdot M \cdot J) & \text{if } k = 1 \\ \sum\limits_{r=1}^{R} \sum\limits_{t=1}^{T} v_r \cdot q_{i,j}(r,t) \cdot p_i^{inc} & \text{if } k > 1 \end{cases} \tag{6}$$

– In the first round, a market value of the resources is not provided to the bidders. Therefore, bidder agents have to formulate the W2P for their *initial bids* with respect to the start-up budget BG^{ini} and their bidding strategy. This is done by calculating a mean bid price that guarantees that the task agents' budget will last for the next L rounds if $M \cdot J$ new sets of XOR-bids are added (cp. line 1 of formula 6).
– In the following rounds, the task agents employ the market values v_r of the resources to determine their W2P. The requested amount of capacity is multiplied by the relevant market value and if the bid is initialized, a factor p^{ini} is included in the calculation so that initial bids may start below or above the current price level of the resource market.

If a bid is rejected, the corresponding W2P is adapted by

$$p_i^{inc} = p^{ini} + (l_i \cdot \Delta p)\,, \tag{7}$$

Round of bid i: $l_i \in \mathbb{N}$
Multiplier increment: $\Delta p \in \mathbb{R}^+$

resulting in the above mentioned value of p^{ini} in round 1 (cp. line 2 of formula 6). To control the price adaption process, an additional price acceleration factor p_i^{inc} is introduced. At each round, P^{inc} is incremented by a constant ΔP. Recalculating the price based on the actual market value v_r results in a faster adoption process. In fact the system quickly reaches a stable state when using this pricing method [19].

Rejected bids are *repeated* with an updated W2P until the bid is accepted, but only for a limited number of rounds. Bids are *discarded* if they are not accepted after L rounds. Furthermore, the task agents' bidding behavior is limited by their budgets. When an agent's budget is exhausted, it formulates no new bids until the budget is refreshed in the next round $k + 1$.[5]

4 Testing Bidding Strategies

This section goes into the details of the agents' *bidding strategies* and their economic motivation. In our simulations we try to find the utility maximizing strategy for the different agent types. The utility function of an agent is defined by:

$$U_a = \frac{\left(\sum_{(i,j)\in B_a} x_{i,j} \cdot \sum_{r=1}^{R} \sum_{t=1}^{T} \ q_{i,j}(r,t)\right)^{\alpha}}{\left(\bar{l_a}\right)^{\beta}} \tag{8}$$

[5] One might consider letting bidding agents get into debt for further bidding or to go finally bankrupt if their debt exceeds a given limit to introduce more realistic behavior, however, this would introduce a new dimension of complexity into the system which is not justified by the gain in 'model realism'.

Utility function of agent a:	U_a	$\in \mathbb{R}^+$
Bids of agent a:	B_a	$\in \{(i,j) \mid i,j \in \mathbb{N}\}$
Accepted bid's time index of agent a:	$\bar{l_a}$	$\in \mathbb{R}^+$
Parameters of the utility function:	α & β	$\in \mathbb{R}^+$

While the amount of acquired ISIP resources has a positive impact but diminishing marginal impact on the agent's utility, the number of periods an agent waits until its bids are accepted has a negative impact. To calculate the decreasing impact of the waiting time, we use a time index $\bar{l_a}$ (the averaged number of periods an agent bids until it has placed a successful bid) and β to adjust the force of the waiting time's impact. The force of the impact of the quantity is defined by α with $0 < \alpha \leq 1$.[6]

Using the utility function we introduce two different agent types: (1) the *quantity maximizer* with $\alpha = 0.5$ and $\beta = 0.01$ and (2) the *impatient bidder* with $\alpha = 0.5$ and $\beta = 1.0$.

– A *quantity maximizer* tries to acquire a high amount of resource capacity. The hypothesis is that this agent follows the strategy of only increasing the bid prices slowly. The economic rationale for this type of proxy agent strategy could be the fact, that it bids for resources required for the fulfillment of an ISIP task that is not time-critical. Referring to the example in section 2, this may be the computation of large time-consuming database jobs on the grid system, that have to be done in a very relaxed time window. A plausible strategy for the bidders is then trying to acquire the required resource capacity bundles at low market values using bids with slightly increasing W2P.
– An *impatient bidder* suffers if he can not use the resources instantaneously and will use an aggressive bidding strategy. This agent has to submit high initial prices, but overpaying will reduce the quantity he can acquire. Moreover, we analyze if a fast inclining pricing strategy can help to further increase the utility of this agent. The economic motivation of this behavior can be a proxy agent that bids for the execution of time-critical tasks in an ISIP provisioning system. A good example for this is the performance of a video conference in the distributed computer system. The conference is scheduled for a narrow time window. The proxy agents have to bid for prompt fulfillment of the resource usage tasks. Therefore it is useful for proxy agents to quickly raise their bids to market level.

The objective of the experiments is to find out the test agent's optimal bidding strategy in competition with the remaining default bidding agents given the two types of utility function (quantity maximizer, impatient bidder) as defined above. Except Δp and p^{ini}, all agents show the same behavior: Beginning with three bundles containing three XOR bids in the first round, both agent types generated three additional bid bundles at each further round k. The task agents

[6] Within this paper the value remains constant $\alpha = 0.5$. We assume a varying impact of the waiting time, which depends on the kind of tasks an agent has to fulfill. See section 2 for examples of these task types.

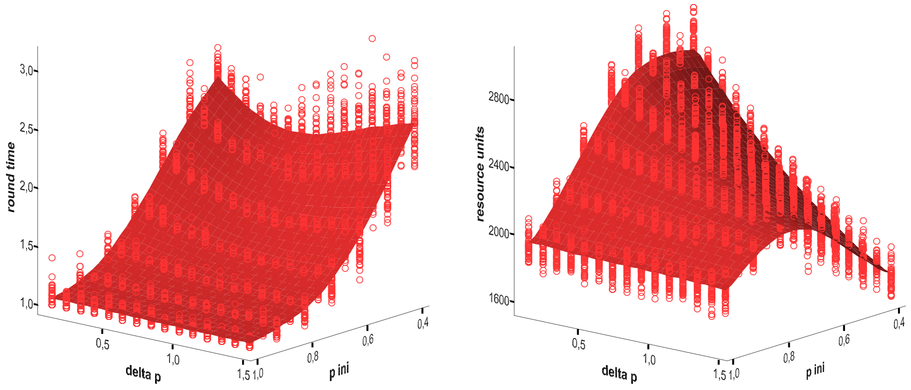


Fig. 4. Mean acceptance time and quantity of resource units for the test bidder with varying price increment Δp and initial price p^{ini}

increase the W2P of rejected bids by a Δp over a maximum of $L = 5$ rounds. The pattern of newly generated bids is identical to the structured BM type described in table 1 ($q_{bmax} = 3, p_{tso} = 0.333, t_{max} = 4$). The auctioneer was able to allocate a maximum load of $q_{max} = 8$ per resource while T was eight units for the CM. The initial budget was set to $BG^{ini} = 200MUs$ for each agent.[7]

For the evaluation of our model we set the number of agents to 4 and the number of bids per agent (M) to 3 (this set results in $I = 12$ bids per round). Three agents use a *default bidding* behavior with constant values of $\Delta p = 0.2$ and $p^{ini} = 0.5$. The value of Δp has to be sufficiently high to guarantee fast price adaptation in the case of resource failures, while a high value of p_{ini} leads to overpaying in the case of low demand.[8] Additionally, a *test agent* with a varying bidding strategy between $\Delta p = 0.1 \ldots 1.5$ and $p^{ini} = 0.4 \ldots 1.0$ is introduced. Table 2 shows the resulting utility of the *test agents*. Figure 4 shows the resource units acquired by the test agent and the averaged bid acceptance time $\bar{l_a}$ of 50 simulation runs for each Δp, p^{ini} combination (steps of 0.1). All task agents within our closed loop economy receive the same budget in each round. Figure 4 illustrates that for small Δp and p^{ini} the highest amount of resource units can be acquired by the test agent. An aggressive strategy with high Δp and p^{ini} leads to a declining amount of acquired resources. While a reduction in acceptance time is mainly achieved by high p^{ini}, increasing Δp only has an impact on average acceptance time if p^{ini} is low. The test agent's utility values (cp. equation 8) are calculated based on the data shown in figure 4. Figure 4 depicts the utility resulting from varying price increment Δp and initial pricing p^{ini}. In the case

[7] The initial budget normally influences the system behavior; however, experiments showed that within certain bounds liquidity does not play a major role due to the price adaption process.

[8] The high variance in the demands of the agents results in a large amount of capacity that could be acquired cheaply. Thus starting with the market price (which is an averaged value) let agents overpay those capacities that have low demand.

Table 2. Utility of the test bidder for quantity maximizing preference $\beta = 0.01$ (upper table) and impatient bidding behavior $\beta = 1.0$ (lower table) for determination of the optimal bidding strategy under varying price increment Δp and initial pricing p^{ini}

	Δp														
p^{ini}	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0	1.1	1.2	1.3	1.4	1.5
0.4	53.05	51.41	50.22	49.14	48.45	47.42	46.47	45.85	45.00	44.09	43.58	42.72	41.79	41.50	40.79
0.5	53.36	52.03	50.84	50.09	49.16	48.45	47.67	46.95	46.20	45.59	44.89	44.23	43.71	42.96	42.56
0.6	53.34	52.46	51.57	50.81	50.22	49.57	48.98	48.32	47.82	47.30	46.71	46.27	45.78	45.24	44.84
0.7	51.93	51.53	50.72	50.30	50.18	49.71	49.36	49.10	48.76	48.09	47.99	47.63	47.36	46.91	46.51
0.8	48.81	48.66	48.69	48.36	48.28	48.10	48.10	47.97	47.68	47.80	47.26	47.29	47.23	46.98	46.87
0.9	45.96	46.08	46.06	46.13	46.02	45.84	45.83	45.85	45.76	45.69	45.69	45.80	45.65	45.56	45.52
1.0	43.80	44.25	43.64	43.68	43.93	43.79	43.85	43.71	43.65	43.81	43.71	43.59	43.57	43.39	43.63

	Δp														
p^{ini}	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0	1.1	1.2	1.3	1.4	1.5
0.4	21.08	21.48	21.96	22.73	22.85	22.42	21.60	21.38	20.30	19.30	18.84	17.84	17.45	16.86	16.64
0.5	23.34	23.91	24.57	25.65	25.72	26.25	26.33	25.67	25.03	24.59	24.04	23.17	22.33	21.49	20.98
0.6	27.73	28.01	28.88	29.80	30.38	30.81	31.43	30.78	30.92	30.74	30.02	30.23	29.36	28.43	28.04
0.7	34.74	34.30	34.10	35.56	36.06	36.06	36.98	37.36	37.56	36.72	36.89	36.56	36.72	36.14	35.57
0.8	39.69	38.95	39.44	40.13	40.56	40.81	40.78	41.51	41.01	41.43	40.24	40.99	40.97	41.15	40.78
0.9	41.77	41.95	41.12	42.13	42.65	41.70	41.96	42.31	42.43	42.34	42.20	42.66	42.39	42.35	42.64
1.0	41.15	42.28	41.50	42.14	41.50	41.49	42.10	41.71	41.97	42.05	41.93	41.93	42.33	41.44	41.83

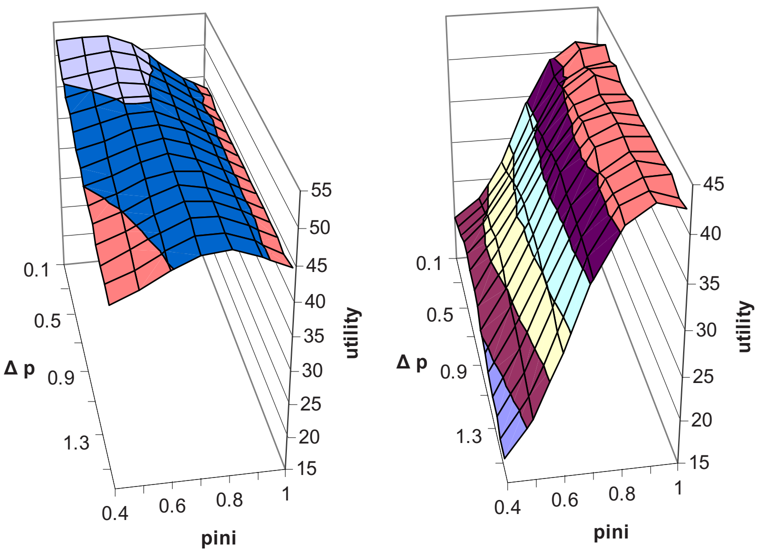


Fig. 5. Utility of the test bidder for quantity maximizing preference $\beta = 0.01$ (left) and impatient bidding behavior $\beta = 1.0$ (right) for determination of the optimal bidding strategy under varying price increment Δp and initial pricing behavior p^{ini}

of a quantity maximizing preference ($\beta = 0.01$), the test agent's utility is high for small initial bids and small price increments with a maximum utility value of $U_a = 53.36$ at $p^{ini} = 0.5$ and $\Delta p = 0.1$ (see table 2 upper part).

It pays off for the quantity maximizer to wait to see if its bids fit into the current allocation at a low price (low increment and initial price). By contrast, the impatient bidder gains low utility from such a strategy (figure 4 right side). The impatient bidder receives the highest utilities by using an initial bid price close to the market value of the resources ($p^{ini} = 0.9$). The price increment in the following round does not have much impact on the acceptance time and the utility of the impatient bidder (see table 2 lower part). For bids exactly at market value utility declines sharply, signaling 'overbidding', which means paying too much for the resources. In the context of the scenario in section 2, this underlines the importance of market value information for achieving good allocations. Our shadow price controlled combinatorial grid provides such information and enables users to acquire resources according to their utility function via proxy-agents. Of cause the impact of the competitors' behavior has to be investigated further.

5 Conclusion

We have presented a simulation environment that enables users to simultaneously allocate resources in a grid-like computer system via proxy-agents. The economically inspired approach uses proxy-agents that try to acquire resource bundles under budget constraints via a mediator that performs a combinatorial auction. Besides solving the winner determination problem by integer programing, this auctioneer also provides resource value information based on shadow prices.

Starting with these prerequisites, various bidding strategies are evaluated while introducing a class of utility functions that is able to express the different levels of time and quantity preferences on the part of the bidders. Two characteristic bidders are investigated: A *quantity maximizer* with low preference for fast bid acceptance and an *impatient bidder* which draws a high utility from fast allocation of the requested resources. While searching for utility maximizing strategies by varying the bidding behavior of the proxy-agents in terms of initial bid price and price increment for rejected bids, two main bidding strategies have been identified. For the quantity maximizing agent it is profitable be patient and to start with low bids, whereas the impatient bidder should avoid 'overbidding' in its tendency to accelerate the resource procurement process.

Two points seem to be important for further research in our combinatorial grid. Firstly, the agents should be able to change their strategies by using learning to explore a wider strategy space. Secondly, individual production functions should be introduced for the agents to come closer to a realistic grid scenario, where the budget received in each round depends on the agents' strategy.

References

1. Foster, I., Kesselman, C., Tuecke, S.: The anatomy of the grid: Enabling scalable virtual organizations. In: Euro-Par '01: Proceedings of the 7th International Euro-Par Conference Manchester on Parallel Processing. Volume 2150 of Lecture Notes in Computer Science. (2001) 1–4
2. Schwind, M.: Dynamic Pricing and Automated Resource Allocation for Complex Information Services - Reinforcement Learning and Combinatorial Auctions. Volume 589 of Lecture Notes in Economics and Mathematical Systems. Springer-Verlag, Berlin (2007)

3. Milgrom, P.: Putting Auction Theory to Work. Cambridge University Press (2004)
4. Buyya, R., Stockinger, H., Giddy, J., Abramson, D.: Economic models for management of resources in peer-to-peer and grid computing. In: Proceedings of the SPIE International Conference on Commercial Applications for High-Performance Computing, Denver, USA (2001)
5. Neumann, D., Holtmann, C., Orwat, C.: Grid-economics. Wirtschaftsinformatik **48**(3) (2006) 206–209
6. Chun, B.N., Buonadonna, P., AuYoung, A., Ng, C., Parkes, D.C., Shneiderman, J., Snoeren, A.C., Vahdat, A.: Mirage: A microeconomic resource allocation system for sensornet testbeds. In: Proceedings of the 2nd IEEE Workshop on Embedded Networked Sensors (EmNetS-II); Sidney, Australia. (2004)
7. AuYoung, A., Chun, B.N., Snoeren, A.C., Vahdat, A.: Resource allocation in federated distributed computing infrastructures. In: Proceedings of the 1st Workshop on Operating System and Architectural Support for the On-demand IT InfraStructure, San Francisco, USA. (2004)
8. Ng, C., Parkes, D.C., Seltzer, M.: Virtual worlds: Fast and strategyproof auctions for dynamic resource allocation. In: Proceedings of the third ACM Conference on Electronic Commerce (EC-2003), San Diego, CA, ACM (2003) 238–239
9. Schnizler, B., Neumann, D., Veit, D., Weinhardt, C.: Trading grid services - a multi-attribute combinatorial approach. European Journal of Operational Research (2006) in print.
10. Foster, I., Jennings, N.R., Kesselman, C.: Brain meets brawn: Why grid and agents need each other. In: Proceedings of the 3rd International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS04), New York, NY (2004) 8–15
11. Schwind, M., Stockheim, T., Rothlauf, F.: Optimization heuristics for the combinatorial auction problem. In: Proceedings of the Congress on Evolutionary Computation CEC 2003. (2003) 1588–1595
12. Parkes, D.C., Ungar, L.H.: Iterative combinatorial auctions: Theory and practice. In: Proceedings of the 17th National Conference on Artificial Intelligence (AAAI-00). (2000) 74–81
13. Fujishima, Y., Leyton-Brown, K., Shoham, Y.: Taming the computational complexity of combinatorial auctions: Optimal and approximate approaches. In: Proceedings of the 16th International Joint Conference on Artificial Intelligence 1999 (IJCAI-99), Stockholm, Sweden. (1999) 548 – 553
14. Vries, S.D., Vohra, R.: Combinatorial auctions: A survey. INFORMS Journal on Computing **15**(3) (2001) 284–309
15. Kwasnica, A.M., Ledyard, J., Porter, D., DeMartini, C.: A new and improved design for multi-objective iterative auctions. Management Science **51**(3) (2005) 419–434
16. Rassenti, J.S., Smith, V.L., Bulfin, R.L.: A combinatorial auction mechanism for airport time slot allocation. The Bell Journal of Economics **13**(2) (1982) 402–417
17. Bjørndal, M., Jørnsten, K.: An analysis of a combinatorial auction. Technical Report 2001-11, Department of Finance and Management Science, Norwegian School of Economics and Business Administration, Bergen, Norway (2001)
18. Schwind, M., Gujo, O.: Using shadow prices for resource allocation in a grid with proxy-bidding agents. In: Proceedings of the 8th International Conference on Enterprise Information Systems (ICEIS 2006), Paphos, Cyprus. (2006) 11–18
19. Schwind, M., Gujo, O., Stockheim, T.: Dynamic resource prices in a combinatorial grid system. In: Proceedings of the IEEE Joint Conference on E-Commerce Technology (CEC'06) and Enterprise Computing, E-Commerce and E-Services (EEE'06), San Francisco, CA. (2006) 356–361

A Comparison of Sequential and Simultaneous Auctions

Shaheen S. Fatima

Department of Computer Science
University of Liverpool, Liverpool L69 3BX, UK
shaheen@csc.liv.ac.uk

Abstract. Sequential and simultaneous auctions are two important mechanisms for buying/selling multiple objects. These two mechanisms yield different outcomes (i.e., different revenues and also different profits to the winning bidders). Hence, both the auctioneer and the bidding agents want to know which mechanism is better for them. Given this, we compare the outcomes for these mechanisms for the following scenario. There are multiple similar objects for sale, each object is sold in a separate auction, and each bidder needs only one object. We use English auction rules and first determine equilibrium bidding strategies for each individual auction for the simultaneous and sequential cases. We do this for both common and private value objects by treating a bidder's information about these values as *uncertain*. We then consider the case where the private and common values have a uniform distribution and compare the two mechanisms in terms of three key properties: a bidder's ex-ante expected profit, the auctioneer's expected cumulative revenue, and the total expected surplus. For both common and private value objects, our study shows the following result. The expected cumulative revenue and the expected total surplus is higher for the sequential mechanism. However, a bidder's exante expected profit depends on the number of objects being auctioned and the number of participating bidders, and it is sometimes higher for the sequential mechanism and sometimes for the simultaneous one.

1 Introduction

Auctions are now being widely used for buying/selling goods on the web. Now, in many cases, there are multiple objects for sale. Multiple objects can be auctioned using a number of mechanisms. Moreover, the bidding behaviour is different for different mechanisms [11]. Hence, the outcome of such auctions (i.e., who wins and at what price) is also different for different mechanisms. Hence, the auctioneer and the bidding agents want to know which mechanism will be the most beneficial to them. Furthermore, the efficiency (i.e., the surplus) is also different for different mechanisms. So it is desirable to find mechanisms that are efficient. However, it is not always possible to have an efficient[1] mechanism. Given this, a key problem in the area of multi-object auctions is to study ad hoc mechanisms and compare them in terms of their solution properties (i.e., revenue, winner's profit, and efficiency). To this end, we study and compare two different mechanisms: *sequential* and *simultaneous* [11]. To date, considerable research effort has been devoted to the study of these mechanisms. But the main limitation of

[1] Auctions for common value objects have been shown to be inefficient [10,7,6].

existing work is that it has studied sequential and simultaneous auctions in isolation and so far there has been very little work on the comparison of these two mechanisms in terms of their outcomes (see Section 5 for details). Our objective is therefore to provide a comparative analysis of sequential (SEQ) and simultaneous (SIM) auctions in an incomplete information setting. We do this for both common value (CV) and private value (PV) objects. This study is important because agents need to decide which of the two mechanisms is better for them. So we compare the mechanisms from the perspective of both the auctioneer and the bidding agents, and also from a global perspective (i.e., in terms of surplus/efficiency).

We use the following setting. There are multiple objects for sale and each bidder needs only one object. Each bidder's valuations (for both CVs and PVs) are identically distributed across the objects to be auctioned but are not perfectly correlated. We first determine equilibrium bidding strategies for each individual auction for the simultaneous and sequential cases by treating each bidder's information about the CVs and PVs as *uncertain*. We do this using the English auction rules. We then consider the case where the private and common values have a uniform distribution and compare the two mechanisms in terms of a bidder's ex-ante expected profit, the seller's expected cumulative revenue, and the expected cumulative surplus. For both CV and PV objects, our study shows the following result. The expected cumulative surplus and the auctioneer's cumulative revenue is higher for the SEQ mechanism. However, the mechanism that generates a higher ex-ante expected profit for the bidders depends on the number of objects being auctioned and the number of participating bidders, and it is sometimes higher for SEQ mechanism and sometimes for the SIM one.

The remainder of the paper is organised as follows. Section 2 describes the auction setting. Section 3 finds the equilibrium strategies for the SEQ and SIM mechanisms. Section 4 compares the two mechanisms for the case where the private and common values have a uniform distribution. Section 5 discusses related literature and Section 6 concludes.

2 The Auction Setting

Single object auctions with both CV and PV elements have been studied in [7]. By changing appropriate parameters, this model can be used to analyse objects which are exclusively CV and those which are exclusively PV. We therefore adopt this basic model and extend it for multiple objects. We begin with a brief overview of the basic model.

Single object. A single object auction is modelled in [7] as follows. There are $n \geq 3$ risk neutral bidders and the object for sale has both CV and PV features. The *CV* (V_1) of the object to the n bidders is equal, but initially the bidders do not know this value. However, each bidder receives a signal that gives an estimate of this CV. Bidder i draws an estimate (v_{i1}) of the object's true value (V_1) from the probability distribution function F with support $[v_L, v_H]$. Although different bidders may have different estimates, the true value (V_1) is the same for all the bidders and is modelled as the average of the bidders'signals: $V_1 = \frac{1}{n}\sum_{i=1}^{n} v_{i1}$. Also, each bidder has a *cost* (which is different for different bidders) and this cost is its *private value*. Let c_{i1} denote bidder i's signal for its private value which is drawn from the distribution function $G(c)$ with support $[c_L, c_H]$ where $c_L \geq 0$ and $v_L \geq c_H$. Cost and value signals are independently and identically distributed across bidders. The term *value* refers to CV and *cost* to PV.

If bidder i wins and pays b, its utility is $V_1 - c_{i1} - b$, where $V_1 - c_{i1}$ is i's surplus. Each bidder bids so as to maximize its utility. Note that i receives two signals (v_i and c_i) but its bid has to be a single number. Hence, in order to determine their bids, bidders need to combine the two signals into a *summary* statistic. This is done as follows. For bidder i, a one-dimensional summary signal, called i's surplus[2], is $S_{i1} = v_{i1}/n - c_{i1}$. This allows i's optimal bids to be determined in terms of S_{i1} (see [7] for details about the problems with two signals and why a one-dimensional surplus is required).

For exclusively CV objects, $c_{i1} = 0$, so i's surplus is $S_{i1} = v_{i1}/n$. For exclusively PV objects, the equilibrium bids are defined in terms of the PVs (which are independently and identically distributed across F). Note that for the PV case, F denotes the distribution function for PVs. We now extend this single object model to $m > 1$ objects.

Multiple objects. There are $m > 1$ similar objects and $n > m$ bidders (Section 3.2 considers the case $n \leq m$). Each bidder needs only one object. Since the m objects are similar, there is a single distribution function (F) for the CV/PV of each object. We now describe the bidders' signals for the CV and PV cases for the two mechanisms. First consider the CV case. For this case, for the SEQ mechanism, each bidder receives its value signal for an auction just before that auction begins. The signal for the jth object is received only after the $(j-1)$ previous auctions have been conducted. Consequently, although the bidders know the distribution functions from which the signals are drawn, they do not know the actual signals for the jth object until the previous $(j-1)$ auctions are over[3]. Our model is therefore more relevant to those scenarios where the objects are *physically distinct* but still *good substitutes*. The m objects are sold in m SEQ auctions conducted using English auction rules. Furthermore, each bidder can win at most one object. The winner for the jth object does not participate in the remaining $m - j$ auctions. Thus, if n agents participate in the first auction, the number of agents for the jth auction is $(n - j + 1)$. For objects $1 \leq j \leq m$ and bidders $1 \leq i \leq n$, let v_{ij} denote the CV for the jth object for bidder i. The true CV of the jth object is: $V_j = \frac{1}{n-j+1} \sum_{i=1}^{n-j+1} v_{ij}$ Also, let S_{ij} denote i's surplus for object j where $S_{ij} = v_{ij}/(n - j + 1)$.

Note that, the CVs of the objects are not correlated. Such correlations occur, if for a bidder (say i) the CV of objects $2 \leq j \leq m$ can be determined on the basis of i's value signal for the first object.

For the SIM mechanism, the m CV objects are sold in m independent English auctions conducted simultaneously. Each bidder goes to only one auction and receives its value signal just before that auction begins (Section 3.3 provides details).

[2] Note that i's true surplus is $V_1 - c_{i1}$ which is equal to $v_{i1}/n - c_{i1} + \sum_{j \neq i} v_{j1}/n$. But since $v_{i1}/n - c_{i1}$ depends on i's signals while $\sum_{j \neq i} v_{j1}/n$ depends on the other bidders' signals, the term 'i's surplus' is also used to mean $v_{i1}/n - c_{i1}$.

[3] This model is an extension of [1]: [1] analyses a sequence of two PV auctions while we analyse $n \geq 2$ auctions for CVs and PVs.

For the PV case, the SEQ and SIM auctions are similar to the CV case except that F now denotes the probability distribution function for the bidders' PVs (instead of CVs).

3 Equilibrium Bidding Strategies

Equilibrium bidding strategies for a single object of the type described in Section 2 have been obtained in [7]. We therefore briefly summarize these strategies and then determine the equilibrium for our m objects case.

3.1 Single Object

We first describe the CV case. For a single object with CV V_1, the equilibrium obtained in [7] is as follows. A bidder's strategy is described in terms of its surplus and indicates how high the bidder should go before dropping out. Since $n \geq 3$, the prices at which some bidders drop out convey information (about the CV) to those who remain active. Suppose k bidders have dropped out at bid levels $b_1 \leq \ldots \leq b_k$. A bidder's (say i's) strategy is described by functions $B_k(x_i; b_1 \ldots b_k)$, which specify how high it must bid given that k bidders have dropped out at levels $b_1 \ldots b_k$ and given that its surplus is $S_i = x_i$. The n-tuple of strategies $(B(\cdot), \ldots, B(\cdot))$ with $B(\cdot)$ defined in Equation 1, constitutes a symmetric equilibrium of the English auction.

$$B_0(x_i) = E(v_i|S_i = x_i)$$
$$B_k(x_i; b_1 \ldots b_k) = \frac{n-k}{n} E(v_i|S_i = x_i) + \frac{1}{n} \sum_{j=0}^{k-1} E(v_i|B_j(x_i; b_1, \ldots, b_j) = b_{j+1}) \tag{1}$$

The intuition for Equation 1 is as follows. Given its surplus and the information conveyed in others' drop out levels, the highest a bidder is willing to go is the expected value of the object, assuming that all other active bidders have the same surplus.

Let f^n denote the first order statistic of the surplus for the n bidders and s^n the second order statistic. For the above equilibrium, the bidder with the highest surplus wins and the winner's expected profit ($EP_w(n)$) is [7]:

$$EP_w(n) = E(f^n) - E(s^n) \tag{2}$$

The expected surplus ($ES(n)$) is the surplus[4] that gets split between the auctioneer and the winning bidder and is:

$$ES(n) = E(V_1) \tag{3}$$

Finally, the difference between $ES(n)$ and $EP_w(n)$ is the expected revenue ($ER(n)$):

$$ER(n) = ES(n) - EP_w(n) \tag{4}$$

[4] Note that this surplus is different from the surplus S_{ij} defined earlier in Section 2. In what follows, the actual meaning of term will be evident from the context in which it is used.

On the other hand, for PV objects, the winner's expected profit ($\overline{EP}_w(n)$) is [18] :

$$\overline{EP}_w(n) = E(\overline{f}^n) - E(\overline{s}^n) \tag{5}$$

The expected surplus ($\overline{ES}(n)$) and the seller's expected revenue ($\overline{ER}(n)$) are:

$$\overline{ES}(n) = E(\overline{f}^n) \tag{6}$$
$$\overline{ER}(n) = E(\overline{s}^n) \tag{7}$$

where $\overline{f}^n$ and $\overline{s}^n$ denote the first and second order statistic for the private value of the object. Both $\overline{f}^n$ and $\overline{s}^n$ are obtained from the probability distribution function F. Note that in the context of exclusively CV objects, F represents the probability distribution function for CVs, while in the context of exclusively PV objects it represents the probability distribution function for PVs. On the basis of the equilibrium for a single object, we determine equilibrium for multiple objects.

3.2 Multiple Sequential Auctions

The m objects are sold in m separate English auctions conducted sequentially one after another. Each bidder needs only one object and draws its CV/PV for an auction just before that auction begins. Since a bidder needs only one object, it participates in the series of auctions until it wins an object. After that, it does not take part in any of the remaining auctions. The English auction rules are as follows. The auctioneer continuously raises the price, and bidders publicly reveal when they withdraw from the auction. Bidders who drop out from an auction cannot re-enter that auction. A bidder's strategy for the jth (for $1 \leq j \leq m-1$) auction depends on its expected profit from the $(m-j)$ future auctions. Since the mth auction is the last one, a bidder's strategy for this auction is the same as the single object case. But the bidding behaviour for the first $(m-1)$ auctions is different from the single object case.

We first obtain the equilibrium bids for CV objects and then extend it to PVs. For the former case, a bidder's strategy for an individual auction depends on its probability of winning any one of the remaining auctions that are yet to be conducted. If the number of bidders for the first auction is n, then let $\beta(y, j, m, n)$ denote a bidder's ex-ante probability of winning the yth (for $j \leq y \leq m$) auction in the series from the jth to the mth auction before the jth one begins. For instance, consider $\beta(1, 1, m, n)$, which is the probability of winning the first auction in the series from the first to the mth auction. Since $\beta(1, 1, m, n)$ is the ex-ante probability (i.e., before the bidders draw their CVs for the first auction), each bidder has equal chances of winning it (i.e., $\beta(1, 1, m, n) = 1/n$) If a bidder wins the first auction then it does not participate in the remaining auctions. Now consider $\beta(2, 1, m, n)$, which is the ex-ante probability that a bidder wins the second auction in the series from the first to the mth auction where $\beta(2, 1, m, n) = (1 - 1/n)(1/(n-1))$. This is because a bidder can win the second auction if it loses the first one. If it wins the second, then it does not participate in the remaining auctions. In the same way we get $\beta(y, 1, m, n)$ and $\beta(y, j, m, n)$ as:

$$\beta(y, 1, m, n) = [\Pi_{k=1}^{y-1}(1 - 1/(n-y+k+1))](1/(n-y+1))$$

$$\beta(y, j, m, n) = [\Pi_{k=j}^{y-1}(1 - 1/(n - y + k + 1))](1/(n - y + 1))$$

Since $m > 1$, the bids for an auction depend not only on that auction but also on a bidder's expected profit from winning one of the future auctions. For CV objects, let $EP_w(j, m, n)$ denote the winner's expected profit for the jth auction in a series of m auctions with n bidders for the first one. Also, let $\alpha(j, m, n)$ denote a bidder's ex-ante expected profit for winning any one auction in the series from the jth (for $1 \leq j \leq m$) to the mth auction where

$$\alpha(j, m, n) = \Sigma_{y=j}^{m}\beta(y, j, m, n)EP_w(y, m, n)$$

A definition for $EP_w(y, m, n)$ will be given in Theorem 2. Note that EP_w with three parameters is used for multiple objects while EP_w with a single parameter for a single object (see Equation 2). Also, note that since there are m objects, $\alpha(m + 1, m, n) = 0$. The equilibrium bids for CV objects depend on α. Theorem 1 characterises the equilibrium for $m > 1$ CV objects. Before presenting the theorem, we introduce some notation. We will denote the first and second order statistic of the surplus for the jth CV auction as f^{n-j+1} and s^{n-j+1} respectively.

Theorem 1. *For CV objects, the n-tuple of strategies $(B(\cdot), \ldots, B(\cdot))$ with $B(\cdot)$ defined in Equation 8 constitutes a symmetric equilibrium for the jth (for $1 \leq j < m$) auction at a stage where k bidders have dropped out:*

$$\begin{aligned} B_0^j(x_{ij}) &= E(v_{ij}|S_{ij} = x_{ij}) - \alpha(j + 1, m, n) \\ B_k^j(x_{ij}; b_1, \ldots, b_k) &= \frac{n - j + 1 - k}{n - j + 1}E(v_{ij}|S_{ij} = x_{ij}) \\ &+ \frac{1}{n - j + 1}\sum_{y=0}^{k-1} E(v_{ij}|B_y(x_{ij}; b_1, \ldots, b_y) = b_{y+1}) - \alpha(j + 1, m, n) \end{aligned} \quad (8)$$

where x_{ij} is bidder i's surplus for the jth object. For the last auction, the equilibrium is as given in Equation 1 with n replaced with $(n - m + 1)$.

Proof. *We consider each of the m auctions by starting with the last auction and reasoning backwards.*

– ***mth auction.*** *For this auction, there are $(n - m + 1)$ bidders. Since this is the last auction, an agent's bids are the same as that for the single object case. Hence, the equilibrium for this auction is the same as that in Equation 1 with n replaced with $(n - m + 1)$. Recall that although the bidders know the distribution (from which the values are drawn) before the first auction begins, they draw the signals for the jth auction only after the $(j - 1)$ earlier auctions end. Hence, for the series from the jth to the mth auction, a bidder's ex-ante expected profit (i.e., the profit computed before the bidders draw their signals for the jth auction), is the same for all participating bidders. Hence from Equation 2 we get:*

$$\alpha(m, m, n) = \frac{1}{n - m + 1}(E(f^{n-m+1}) - E(s^{n-m+1}))$$

This is because all the $(n-m+1)$ agents have ex-ante identical chances of winning.

- $(m-1)$***th auction.*** *For this auction, a bidder bids b if $(V_{m-1} - b \geq \alpha(m,m,n))$ or $b \leq V_{m-1} - \alpha(m,m,n)$. Hence, an equilibrium for the $(m-1)$th auction is obtained by substituting $j = m-1$ in Equation 8. The difference between the bids for the single object case and the $(m-1)$th auction is α (see Equations 1 and 8). Since the bids decrease by $\alpha(j+1,m,n)$, the winner's profit now increases to:*

$$EP_w(m-1,m,n) = E(f^{n-m+2}) - E(s^{n-m+2}) + \alpha(m,m,n)$$

- ***First*** $(m-2)$ ***auctions.*** *Generalising Equation 9 to the first $(m-1)$ auctions, we get the winner's expected profit ($EP_w(j,m,n)$) as:*

$$EP_w(j,m,n) = E(f^{n-j+1}) - E(s^{n-j+1}) + \alpha(j+1,m,n)$$

  Consequently, a bidder's equilibrium bid for the jth auction is obtained by discounting the single object bid by $\alpha(j+1,m,n)$. Hence, we get the equilibrium bids of Equation 8. □

Let $ER(j,m,n)$ denote the expected revenue from the jth auction in a series of m CV auctions with n bidders for the first one. Also, let $ES(j,m,n)$ denote the expected surplus for the jth auction. The following theorem characterises the outcome for CV auctions.

Theorem 2. *For the jth (for $1 \leq j \leq m$) auction, the winner's expected profit ($EP_w(j,m,n)$), the expected surplus ($ES(j,m,n)$), and the expected revenue ($ER(j,m,n)$) are:*

$$\begin{aligned} \forall_{j=1}^{m-1} EP_w(j,m,n) &= E(f^{n-j+1}) - E(s^{n-j+1}) + \alpha(j+1,m,n) \\ EP_w(m,m,n) &= E(f^{n-m+1}) - E(s^{n-m+1}) \\ \forall_{j=1}^{m} ES(j,m,n) &= E(V_j) \\ \forall_{j=1}^{m} ER(j,m,n) &= ES(j,m,n) - EP_w(j,m,n) \end{aligned} \tag{9}$$

Proof. *For the jth $(1 \leq j < m)$ auction, the bids in Theorem 1 are similar to those in Equation 1 (for the single object case), except that each bid in the former case is obtained from the corresponding bid in the latter by shifting the latter by the constant $\alpha(j+1,m,n)$. Since $\alpha(j+1,m,n)$ is the same for all participating bidders, the relative positions of bidders for each of the m auctions remains the same as that for the corresponding single object case. Also, since the bids are discounted by $\alpha(j+1,m,n)$, the winner's expected profit increases by the same amount. So we get:*

$$EP_w(j,m,n) = E(f^{n-j+1}) - E(s^{n-j+1}) + \alpha(j+1,m,n)$$

For the last auction, the winner's profit is the same as the single object case. The surplus ($ES(j,m,n)$) that gets split between the auctioneer and the winning bidder is obtained from Equation 3 as $ES(j,m,n) = E(V_j)$. The revenue for the jth auction is the difference between $ES(j,m,n)$) and $EP_w(j,m,n)$:

$$\forall_{j=1}^{m} ER(j,m,n) = ES(j,m,n) - EP_w(j,m,n) \tag{10}$$

□

We now find the cumulative revenue, the cumulative surplus, and a bidder's ex-ante expected profit from all the m auctions. For ($n > m$), let $X_{seq}(m,n)$ denote the cumulative revenue and $Y_{seq}(m,n)$ a bidder's exante profit from all the m auctions where $X_{seq}(m,n) = \Sigma_{j=1}^{m} ER(j,m,n)$ and $Y_{seq}(m,n) = \alpha(1,m,n)$. In general (i.e., for $n > m$ or $n \leq m$), the cumulative revenue ($ECR_{seq}(m,n)$) is:

$$ECR_{seq}(m,n) = \begin{cases} 0 & \text{if } m = 0, n = 0, \text{ or } n = 1 \\ X_{seq}(m,n) & \text{if } n > m \\ X_{seq}(n-1,n) & \text{if } n \leq m \end{cases}$$

This is because if $n \leq m$, then the number of bidders for the first $n - 1$ auctions is at least 2. For the nth auction there is only one bidder and so it wins the object for nothing resulting in a revenue of zero. For all the remaining $m - n$ auctions, there are no bidders and so the revenue is zero again. Also, let $EEP_{seq}(m,n)$ denote a bidder's ex-ante expected profit for all the m auctions where

$$EEP_{seq}(m,n) = \begin{cases} E(V) & \text{if } n = 1 \\ Y_{seq}(m,n) & \text{if } n > m \\ Y_{seq}(n-1,n) + \beta(n,1,n,n)E(V) & \text{if } n \leq m \end{cases}$$

This is because if ($n \leq m$) for the first $n-1$ auctions there are at least two bidders. But for the nth auction, since there is only one bidder, it gets the object for nothing. For this auction, the winner's expected profit is the expectation of the CV. Since the CVs of all the objects are drawn from a single distribution function (F), we drop the subscript in $E(V)$. Finally, the expected cumulative surplus is:

$$ECS_{seq}(m,n) = \begin{cases} mE(V) & \text{if } n \geq m \\ nE(V) & \text{if } n < m \end{cases}$$

We now turn to PV objects. For these objects, we denote the first and second order statistic of the PV for the jth auction as $\overline{f}^{n-j+1}$ and $\overline{s}^{n-j+1}$ respectively. For these objects, a bidder's bid for an auction is obtained by subtracting its ex-ante expected profit for the future auctions from its bid for the current one (this is similar to the CV case analysed in Theorem 1). Hence, for the jth auction, the winner's expected profit (denoted $\overline{EP}_w(j,m,n)$) is:

$$\forall_{j=1}^{m-1} \overline{EP}_w(j,m,n) = E(\overline{f}^{n-j+1}) - E(\overline{s}^{n-j+1}) + \overline{\alpha}(j+1,m,n) \quad (11)$$

$$\overline{EP}_w(m,m,n) = E(\overline{f}^{n-m+1}) - E(\overline{s}^{n-m+1}) \quad (12)$$

where $\overline{\alpha}(j,m,n)$ is a bidder's ex-ante expected profit for winning any one auction in the series from the jth (for $1 \leq j \leq m$) to the mth auction and is:

$$\overline{\alpha}(j,m,n) = \Sigma_{y=j}^{m} \beta(y,j,m,n)\overline{EP}_w(y,m,n)$$

The expected revenue ($\overline{ER}(j,m,n)$) and the expected surplus ($\overline{ES}(j,m,n)$) are:

$$\forall_{j=1}^{m-1} \overline{ER}(j,m,n) = E(\overline{s}^{n-j+1}) - \overline{\alpha}(j+1,m,n) \quad (13)$$

$$\overline{ER}(m,m,n) = E(\overline{s}^{n-m+1}) \quad (14)$$

$$\forall_{j=1}^{m} \overline{ES}(j,m,n) = \overline{ER}(j,m,n) + \overline{EP}_w(j,m,n) \quad (15)$$

For $n > m$, let $\overline{X}_{seq}(m,n)$ denote the expected cumulative revenue for all the m objects and $\overline{Y}_{seq}(m,n)$ a bidder's ex-ante expected profit from all the m auctions where $\overline{X}_{seq}(m,n) = \Sigma_{j=1}^{m} \overline{ER}(j,m,n)$ and $\overline{Y}_{seq}(m,n) = \overline{\alpha}(1,m,n)$. In general, (i.e., for $n > m$ or $n \leq m$), the cumulative revenue ($\overline{ECR}_{seq}(m,n)$) is:

$$\overline{ECR}_{seq}(m,n) = \begin{cases} 0 & \text{if } m = 0,\, n = 0, \text{ or } n = 1 \\ \overline{X}_{seq}(m,n) & \text{if } n > m \\ \overline{X}_{seq}(n-1,n) & \text{if } n \leq m \end{cases}$$

and a bidder's ex-ante expected profit ($\overline{EEP}_{seq}(m,n)$) is:

$$\overline{EEP}_{seq}(m,n) = \begin{cases} E(\overline{f}^{1}) & \text{if } n = 1 \\ \overline{Y}_{seq}(m,n) & \text{if } n > m \\ \overline{Y}_{seq}(n-1,n) + \beta(n,1,n,n)E(\overline{f}^{1}) & \text{if } n \leq m \end{cases}$$

Finally, the expected cumulative surplus is:

$$\overline{ECS}_{seq}(m,n) = \begin{cases} \sum_{j=1}^{m} E(\overline{f}^{n-j+1}) & \text{if } n \geq m \\ \sum_{j=1}^{n} E(\overline{f}^{n-j+1}) & \text{if } n < m \end{cases}$$

3.3 Multiple Simultaneous Auctions

As before, we have $m > 1$ similar objects, n risk-neutral bidders, and each bidder needs only one object. Given this, the SIM auctions game is played as follows. The m objects are sold in m different auctions conducted simultaneously and independently of each other. Each bidder randomly selects one of the m auctions and bids in it. It is obvious that after a bidder selects an auction, its equilibrium bids are the same as that for the single object scenario of Section 3.1. This game has been analysed in [17] for private value objects for the limiting case where both m and n tend to infinity. The analysis is done by assuming complete information. Here, we extend this analysis to a more realistic case where both m and n are finite, the objects are either CV or PV, and there is uncertainty about these values. For this SIM auctions game, the equilibrium outcome depends on how the n bidders arrive at the m auctions. We let T_m^n denote the number of different ways in which n bidders can be distributed between m auctions where

$$T_m^n = \begin{cases} 1 & \text{if } n = 0 \text{ or } m = 1 \\ \Sigma_{k=0}^{n} C(n,k) & \text{if } m = 2 \\ \Sigma_{k=0}^{n} C(n,k) T_{m-1}^{n-k} & \text{if } m > 2 \end{cases}$$

For m SIM auctions, the probability that k bidders arrive at an auction ($P(m,k)$) is:

$$P(m,k) = (C(n,k)T_{m-1}^{n-k})/T_m^n$$

Consider any one auction. The number of bidders for this auction can vary between 0 and n. If no bidders arrive at the auction, the object remains unsold. If only one bidder arrives, it gets the object for nothing. If $k \geq 2$ bidders arrive, then the expected revenue is greater than zero. Consider first CV objects. For these objects, the expected cumulative revenue from all the m auctions ($ECR_{sim}(m,n)$) is:

$$ECR_{sim}(m,n) = \Sigma_{k=0}^{n} P(m,k)[ER(k) + ECR_{sim}(m-1,n-k)]$$

Also, for an individual auction, the winner's expected profit is:

$$EP_w(m,n) = P(m,1)[E(V)] + \Sigma_{k=2}^{n} P(m,k)[E(f^k) - E(s^k)]$$

and hence a bidder's ex-ante expected profit ($EEP_{sim}(m,n)$) is:

$$EEP_{sim}(m,n) = P(m,1)[E(V)] + \Sigma_{k=2}^{n} P(m,k)(1/k)[E(f^k) - E(s^k)]$$

Finally, the total expected surplus (denoted $ECS_{sim}(m,n)$) is:

$$ECS_{sim}(m,n) = \Sigma_{k=1}^{n} P(m,k)[ES(k) + ECS_{sim}(m-1,n-k)]$$

Now consider PV objects. For these objects, the expected cumulative revenue is:

$$\overline{ECR}_{sim}(m,n) = \Sigma_{k=0}^{n} P(m,k)[\overline{ER}(k) + \overline{ECR}_{sim}(m-1,n-k)]$$

Also, for an auction, the winner's expected profit is:

$$\overline{EP}_w(m,n) = P(m,1)[E(\overline{f}^1)] + \Sigma_{k=2}^{n} P(m,k)[E(\overline{f}^k) - E(\overline{s}^k)]$$

and hence a bidder's ex-ante expected profit (denoted $\overline{EEP}_{sim}(m,n)$) is:

$$\overline{EEP}_{sim}(m,n) = P(m,1)[E(\overline{f}^1)] + \Sigma_{k=2}^{n} P(m,k)(1/k)[E(\overline{f}^k) - E(\overline{s}^k)]$$

Finally, the total expected surplus (denoted $\overline{ECS}_{sim}(m,n)$) is:

$$\overline{ECS}_{sim}(m,n) = \Sigma_{k=1}^{n} P(m,k)[\overline{ES}(k) + \overline{ECS}_{sim}(m-1,n-k)]$$

4 A Comparison of the Two Auction Mechanisms

We now compare the two mechanisms in terms of three key properties: the expected cumulative revenue, the expected cumulative surplus, and a bidder's ex-ante expected profit. To do this, we consider the uniform distribution function for both PVs and CVs. Recall that the m objects are similar. Hence we have a single probability density function (pdf) for the values of all the m objects. We first describe the experimental setting for the CV case and then for PVs. For the CV case, we let $f(x)$ and $g(x)$ denote the pdfs for value and surplus respectively where $f(x)$ is uniformly distributed over $[100, 200]$:

$$f(x) = \begin{cases} 1/100 & \text{if } 100 \leq x \leq 200 \\ 0 & \text{otherwise} \end{cases}$$

Thus the cumulative distribution function (cdf) for the CVs is $F(x) = \frac{x}{100} - 1$ for $100 \leq x \leq 200$. Consequently, for n bidders, the pdf for surplus is uniformly distributed as:

$$g(x) = \begin{cases} n/100 & \text{if } \frac{100}{n} \leq x \leq \frac{200}{n} \\ 0 & \text{otherwise} \end{cases}$$

The cdf for the surplus is therefore $G(x) = \frac{nx}{100} - 1$ for $\frac{100}{n} \leq x \leq \frac{200}{n}$. For the PV case, $f(x)$ is the pdf for the PV of each of the m objects.

For this setting, we find the cumulative revenue, the cumulative surplus, and a bidder's ex-ante expected profit for the SEQ and SIM mechanisms. In order to do this, we find the first and second order statistics for $F(x)$ and $G(x)$ as follows. From a distribution with cdf $G(x)$, if n random samples are drawn, then the expectation of the first and second highest order statistic (denoted $E(f^n)$ and $E(s^n)$ respectively) of these, between limits $\overline{x}$ and $\underline{x}$, is [4]:

$$E(f^n) = n \int_{\underline{x}}^{\bar{x}} xG(x)^{n-1} g(x) dx \tag{16}$$

$$E(s^n) = n(n-1) \int_{\underline{x}}^{\bar{x}} xG(x)^{n-2} [1 - G(x)] g(x) dx \tag{17}$$

Hence, $E(\overline{f}^n)$ and $E(\overline{s}^n)$ are obtained by replacing G and g in Equations 16 and 17 with F and f respectively. Using these equations, we first study the relation between the expected cumulative revenues for the two mechanisms by varying the number of bidders (n) and the number of objects (m). Figures 1(a) and 1(b) depict the relation for CV and PV objects respectively, for $m = 3$, $m = 4$, and $m = 5$. We then study the relation between the expected cumulative surplus for the two mechanisms (see Figures 2(a) and 2(b)). Finally, we study the relation between a bidder's ex-ante expected profit for the two mechanisms (see Figures 3(a) and 3(b)). As seen in the figures, for both CV and PV objects, the expected cumulative revenue and the expected cumulative surplus is higher for the SEQ mechanism than the SIM one. However, a bidder's ex-ante expected profit is sometimes higher for the SEQ mechanism and sometimes for the SIM one. For $5 < m \leq 50$, the SEQ mechanism generated a higher cumulative revenue, a higher cumulative surplus, and also a higher exante expected profit for the bidders. Furthermore, for both CV and PV objects, as m increased beyond 5, the three differences (i.e., $ECR_{seq} - ECR_{sim}$, $ECS_{seq} - ECS_{sim}$, and $EEP_{seq} - EEP_{sim}$) were not only positive but increased with m.

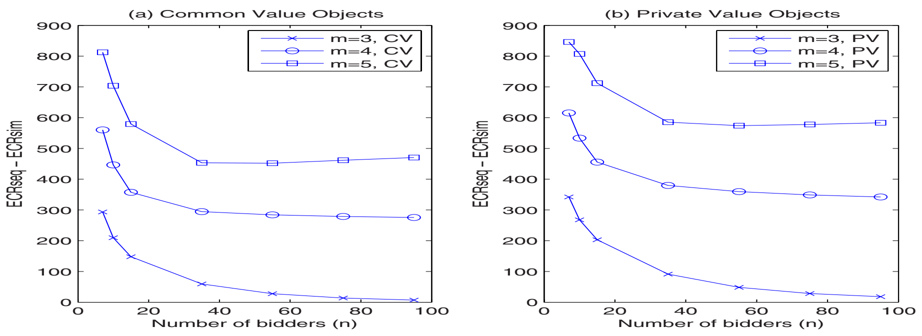


Fig. 1. Difference between the expected cumulative revenues for the two mechanisms

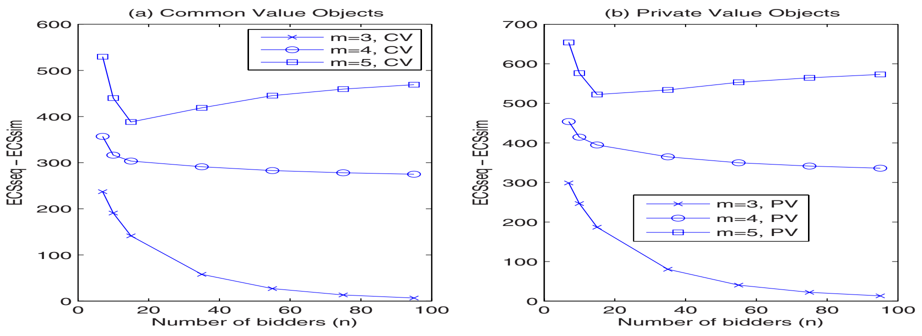


Fig. 2. Difference between the expected cumulative surplus for the two mecahnisms

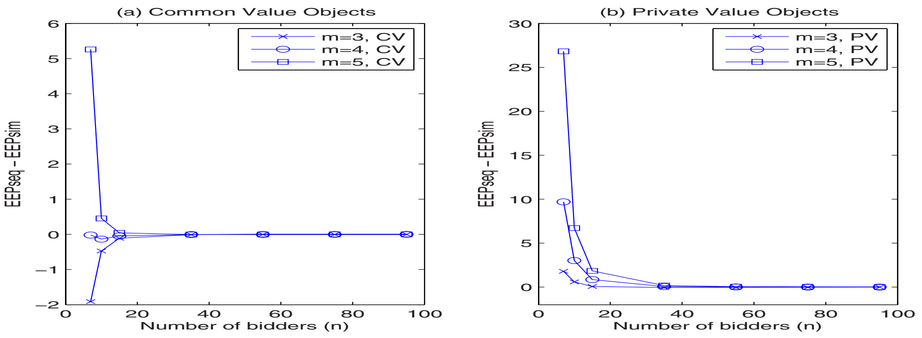


Fig. 3. Difference between a bidder's ex-ante expected profit for the two mecahnisms

In summary, from an auctioneer's perspective, the SEQ mechanism is better because it generates a higher revenue. From a global perspective too, the SEQ mechanism is better (because it generates a higher expected cumuative surplus). However, from a bidder's perspective, the ex-ante expected profit depends on the number of participating bidders and the number of objects being auctioned, and it is sometimes higher for the SEQ mechanism and sometimes for the SIM one.

5 Related Work

Although there has been much work on SEQ auctions [14,19,13,12,1,6], there has been relatively less work on the comparison of different auction mechanisms using a game theoretic approach[5]. For instance, [15] provided a comparison of the revenues generated by two multi-object mechanisms: one offers each object in a separate ascending bid auction and the other offers the bundle of objects as a single item in an ascending bid auction. The key conclusion of this work is that the mechanism that gives a higher

[5] Some work on the comparison of auction mechanisms has been done using other approaches. For example, [2] uses an evolutionary approach while [16] uses an approach based on learning.

revenue depends on the number of bidders. [9] compared SEQ and SIM auctions for two PV objects using first-price sealed bid rules. In this model, each bidder draws its signals for both objects before the first auction begins. For the SEQ case, the bids for the first auction are submitted. These bids are then unsealed and the price is publicly announced. Then bidders submit bids for the second auction. For the SIM case, all the objects are sold in a lot and all the bids on all the objects are submitted before any are unsealed and publicly announced. The key result is that, depending on the pdf for the signals, the revenue is sometimes higher for SEQ sale and sometimes it is higher for SIM sale. This work also shows that if the revenue is higher for the SEQ (SIM) case, then a bidder's ex-ante profit is lower for the SEQ (SIM) case. So the seller and the bidders have conflicting preferences over their choice of a mechanism. The key differences between this work and ours are as follows. The first difference is in terms of the number and type of objects. We consider $m \geq 2$ CV and PV objects while [9] considers two PV objects. Second, in [9], the bidders receive their signals for both objects before the first auction begins. In our model, there is a single pdf for the CVs and PVs of all the objects and each bidder receives its signals for an object before the auction for that object begins (like in [14,1]). Our model is therefore more relevant to those scenarios where the objects are physically distinct but still good substitutes. The final difference is in terms of results. Our analysis shows that the auctioneer and the bidders may have similar or conflicting preferences over their choice of a mechanism, while [9] shows that their preferences always conflict.

Other work on comparison includes [8,5]. [8] study *heuristic* bidding strategies and show that such strategies are optimal for SEQ auctions but not for SIM ones. [5] is more akin to our work and provides a comparative study of the SEQ and SIM auctions described here. However, the key difference between [5] and our present work is that, for the former, each object has both common and private value elements, while the latter considers objects that either exclusively CV or exclusively PV.

Finally, FCC auctions are known to be more efficient than SEQ auctions [3]. A key difference between the FCC auctions (which are SIM ascending auctions) and the SIM auctions we analyse is that, for the latter, there is a separate auction for each object and the bidders are randomly spilt across these auctions. But for the former, all the objects are sold in a lot in which all the bidders participate. Our work shows that although having a separate auction for each object and splitting the bidders across them reduces both revenue and auction efficiency, it can increase a bidder's ex-ante expected profit.

6 Conclusions and Future Work

This paper analyses two key auction mechanisms for multiple objects: *sequential* and the *simultaneous*. This analysis is done for both common and private value objects by treating a bidder's information about these values as uncertain. We used the English auction rules and determined equilibrium for the two mechanisms. We then considered the case where the private and common values have a uniform distribution and compared the mechanisms in terms of their expected cumulative revenues, their expected cumulative surplus, and a bidder's ex-ante expected payoff. For both common and private values, our study shows that, the sequential mechanism is better both from the auctioneer's and also from a global perspective. However, from a bidder's perspective, the choice of a mechanism depends on the number of objects and the number of bidders.

In future, we will extend our analysis to the case where each bidder needs multiple objects. Also, we modelled a bidder's uncertainty about the CVs and PVs with the uniform distribution. In future, we will generalise our results to other types of distributions. Finally, we compared the SEQ mechanism with one specific SIM mechanism – the one in which each bidder bids in a single randomly chosen auction. In future, we will extend the comparison to a SIM mechanism in which the bidders bid in a randomly chosen subset of auctions since doing so increases their chances of winning an object.

References

1. D. Bernhardt and D. Scoones. A note on sequential auctions. *American Economic Review*, 84(3):653–657, 1994.
2. D. Cliff. Evolution of market mechanisms through a continuous space of auction types. Technical Report HPL-2001-326, HP Laboratories, Bristol, 2001.
3. P. Cramton. Simultaneous ascending auctions, http://www.cramton.umd.edu/papers2000–2004, 2004.
4. H. David. *Order Statistics*. Wiley, New York, 1969.
5. S. S. Fatima. A comparative study of sequential and simultaneous auctions. In Submission, 2006.
6. S. S. Fatima, M. Wooldridge, and N. R. Jennings. Sequential auctions for objects with common and private values. In *Fourth International Conference on Autonomous Agents and Multi-Agent Systems*, pages 635–642, Utrecht, Netherlands, 2005.
7. J. K. Goeree and T. Offerman. Competitive bidding in auctions with private and common values. *The Economic Journal*, 113(489):598–613, 2003.
8. A. Greenwald and J. Boyan. Bidding under uncertainty. In *Twentieth Conferene on Uncertainty in Artificial Intelligence*, pages 209–216, 2004.
9. D. B. Hausch. Multi-object auctions: sequential vs. simultaneous sales. *Management Science*, 32(12):1599–1610, 1986.
10. P. Jehiel and B. Moldovanu. Efficient design with interdependent valuations. *Econometrica*, 69:1237–1259, 2001.
11. V. Krishna. *Auction Theory*. Academic Press, 2002.
12. R. P. McAfee and D. Vincent. The declining price anomaly. *Journal of Economic Theory*, 60:191–212, 1993.
13. P. Milgrom and R. J. Weber. A theory of auctions and competitive bidding II. In *The Economic Theory of Auctions*. Edward Elgar, Cheltenham, U.K, 2000.
14. A. Ortega-Reichert. Models of competitive bidding under uncertainty. Technical Report 8, Stanford University, 1968.
15. T. Palfrey. Bundling decisions by a multiproduct monopolist with incomplete information. *Econometrica*, 51:463–484, 1983.
16. D. Pardoe and P. Stone. Developing adaptive auction mechanisms. *SIGecom Exchanges*, 5(3):1–10, 2005.
17. R.Engelbrecht-Wiggans and R. J. Weber. An example of a multi-object auction game. *Management Science*, 25(12):1272–1277, 1979.
18. W. Vickrey. Counterspeculation, auctions and competitive sealed tenders. *Journal of Finance*, 16:8–37, 1961.
19. R. J. Weber. Multiple-object auctions. In R. Engelbrecht-Wiggans, M. Shibik, and R. M. Stark, editors, *Auctions, bidding, and contracting: Uses and theory*, pages 165–191. New York University Press, 1983.

A Market-Pressure-Based Performance Evaluator for TAC-SCM*

Brett Borghetti, Eric Sodomka, Maria Gini, and John Collins

Dept of Computer Science and Engineering, University of Minnesota,
200 Union St SE, Minneapolis, MN 55455
{borg,sodomka,gini,jcollins}@cs.umn.edu
http://www.cs.umn.edu/~borg

Abstract. We propose a novel method to characterize the performance of autonomous agents in the Trading Agent Competition for Supply Chain Management (TAC-SCM). We create a suite of testing tools that reduce the variability of TAC-SCM games, make them replayable, and generate specific market conditions under which autonomous trading agents can be tested. Using these tools, we show how developers can inspect their agents to reveal and correct undesirable behaviors that might otherwise have gone undiscovered. We also discuss how these tools can be used to improve overall trading agent performance in future competitions.

1 Introduction

One of the most prominent proving grounds for current research in autonomous trading agents is the Trading Agent Competition for Supply Chain Management (TAC-SCM) [1]. In this yearly international event, autonomous agents battle for supremacy in a simulation where the highest profit-earning agent wins. In TAC-SCM, agents make all the decisions to run a virtual computer-manufacturing operation. They negotiate to purchase parts from suppliers, optimize their assembly lines, and sell by auction their products to customers. They manage parts and product inventories, minimize costs, optimize revenue, and try to out-earn their competitors.

Agents compete against 5 other adversaries in each game. Different combinations of competitors, in addition to the randomness in the game, cause different market conditions to arise. An agent must perform well under many different market conditions to succeed.

This paper describes one of several tools that our research team created to improve the TAC-SCM agent software developer's ability to test and evaluate their software. Our goal in this work is to provide methods for developers to test whether the modifications they make to their agents will improve their

* Partial funding provided by NSF under grant IIS-0414466. We would like to thank the entire University of Minnesota Trading Agent Research Group for their help and support with this effort.

agent's chance of success in the TAC-SCM competition. We explored two key facets of testing: (1) variability reduction, and (2) manipulation of the market characteristics.

We modified the TAC-SCM server to give a developer the ability to control the extent of the randomness in the suppliers and the customers. We also generated a limited replay capability so different combinations of agents could experience the identical sequence of random processes in the suppliers and customers in successive trials.

To manipulate the environment, we developed a pair of benchmark agents that control market conditions to simulate specific levels of supply and demand in the market. Any team wishing to evaluate their agent can use these stand-alone market manipulator agents to create a configureable level of pressure in the marketplace. The benchmark agents do not require alteration of the game server or the agent being examined. They control the supply and demand characteristics of a game by buying parts and selling computers at prices that generate the desired demand and supply in the marketplace.

The remainder of this paper is organized as follows. In section 2 we describe other research efforts in this area. Then we discuss some of the challenges we face and provide an explanation of our approach in section 3 and our experiments in section 4. Section 5 and 6 provide our conclusions and plans for future work.

2 Related Work

TAC-SCM allows research teams to develop trading agents and compare their relative performance in a complex, standardized environment [1]. Several teams developed methods of analyzing agent performance. The University of Southampton team examined running variations of their competitive agent with different risk strategies for customer pricing [2] to show that their competitive agent made the highest profit among the variations. They also developed a set of controlled experiments for measuring the relative performance of several variations of procurement strategies [3], including Short Term Planning, Long Term Planning, and Mixed strategy. The University of Michigan team analyzed post-competition performance of the TAC-SCM 2004 finalists and explored relationships between total profit and other measurements of performance [4]. The team at the University of Texas at Austin focused on comparing relative performance of variations of their TacTex [5], [6], [7] agent. Their baseline for each suite of controlled tests was a consistent set of other competitor agents downloaded from the SICS agent repository[1].

While each of these teams compared relative agent performance no one developed a stable, universal benchmarking environment in which to characterize agent performance against a standard reference. We felt in order for the field to advance, we must explore this region of performance analysis.

[1] http://www.sics.se/tac/showagents.php

3 Approach

There are two key challenges in testing TAC-SCM agents. The first is that after a team makes a slight modification or change of parameters, due to the inherent randomness in the game, the team must run a significant number of simulations to determine statistically whether or not the changes have improved the agent and ensure that no undesired side effects have been introduced. The second challenge is that outside of an actual competition, it is hard to recreate the competitive effects of multiple agent interactions on the market. We explore these two challenges in two separate approaches: (1) reducing the variability of the gamespace and (2) generating specific market conditions for focused observation of agent behavior.

3.1 Reducing Variability of the Gamespace

In an environment such as the trading agent competition, there are many variables which can influence total performance of a given agent. These include the random variables such as the daily capacity of the suppliers and the demand of the customers, as well as the effects that other agents playing in the game have on the environment.

Since each game simulation takes an hour of real time (and usually 7 processors to run 6 agents and the server), running a large number of games after each minor change to the agent code may be unfeasible. The approach we use in this research allows us to perform a small number of tests to indicate whether the agent is behaving as desired before the large battery of tests is run. While the approach may be statistically inconclusive for determining absolute performance improvement, the additional variability reduction that these methods provide will improve the accuracy of the regression test process.

We explored two methods of variability reduction: 1.) minimizing the randomness; and 2.) controlling the random number generator seed to enable repeatable games. While we employed some of the variability reduction techniques described below to facilitate our market manipulation experiments, a full treatment of variability reduction techniques will be presented in a future paper.

In our first set of experiments, we wanted to reduce the effect of random variables in the game. To minimize the randomness of a given game, we altered the minimum and maximum intervals of the random variables in the TAC server configuration file. In our experiments, we chose to lock the normally random variables into neutral values representing the default values for each of the characteristics (as listed in [1]). To do this, we altered the minimum and maximum intervals that variables were allowed to hold in the server configuration files. We set default values for all $[min, max]$ intervals that are configurable within the server. Thus our new settings for each interval are $[x, x]$ where $x = \frac{(min+max)}{2}$.

These changes allow us to observe two important aspects of our agent's performance at the locked default server values for supplier and customer behavior. First, since we know the exact supply and demand levels in the game, we can estimate how much of that demand our agent should be attempting to meet, and we can determine if the agent is functioning properly under steady-state conditions. Second, by locking the game's random variables on the customer and supplier side, we can observe steady state performance differences between different versions of our agent during the development cycle. This facilitates the regression test process for the agent.

While the variability reduction method discussed above does allow us controlled observations of the steady-state performance of an agent, it does not allow an agent to experience and respond to the ebb and flow of supply and demand that would normally occur in a supply chain marketplace. The second method we explored to reduce variability preserves some of the fluctuating aspects of the marketplace while still allowing a strongly-controlled environment for testing agent performance.

In the second method we created a system that allows the exact sequence of supplier and consumer random variables in a game to be replayed. First, we identified and separated the usages of the TAC servers' random number generator into two categories: server-side independent random processes and server-side tie breaking. Our goal was to control the order of the random numbers generated on server-side random processes without changing the behavior of the client agents. We modified the server by disconnecting the server side random processes and running them with their own configurable random seeds. Doing this ensured game repeatability for the random processes used by the server while still maintaining a separate true random number generation capability for agent behavior.

3.2 Controlling the Market Conditions

The main focus of this paper is to discuss the techniques we used to test an agent experiencing the potential competitive pressure of yet-to-be-seen competitors. To generate various competitive effects we developed two new agents, the *Market Relief* or "do nothing" agent, and the *Market Pressure* agent. The Market Relief Agent occupies one or more of the 6 slots in a TAC simulation without making any financial transactions. This agent provides relief to the market from the perspective of the other agents in two ways. First, it reduces demand on the suppliers which leads to a lowering of supplier prices. Second, it reduces available supply for the customers which causes customers to pay more for computers from other agents.

Designing the Market Relief Agent was relatively simple. We used the example agent code available for download from the Swedish Institute of Computer Science (SICS) Trading Agent Competition website.[2] We modified several areas in the code where it made decisions regarding which customers' request for quotes (RFQs) the agent should make offers on. By setting those decisions to never bid on RFQs, we effectively disabled the agent. Since it never made any bids to customers and it was designed as a build-to-order agent, it never ordered any supplies. When used in a testing environment with other agents, this agent reduces demand on the suppliers and reduces available supply of products for

[2] http://www.sics.se/tac/page.php?id=16

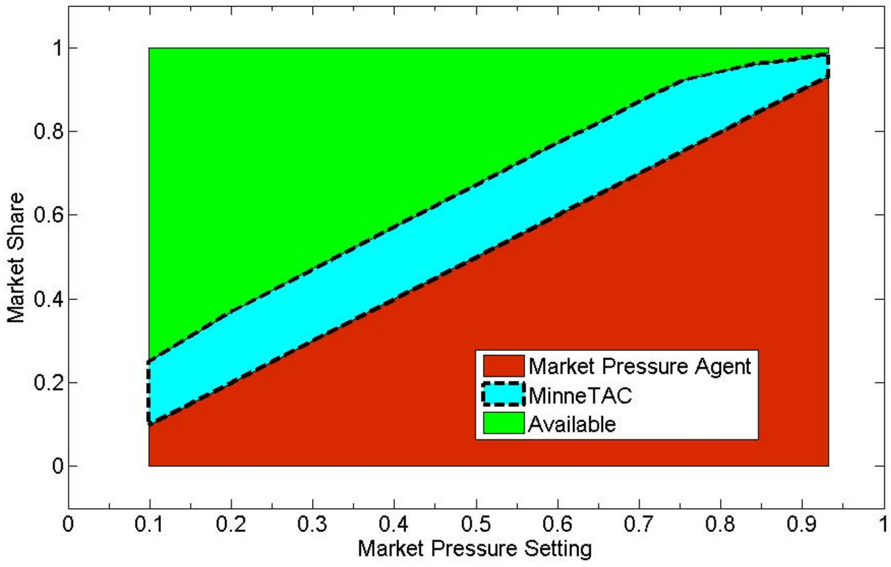


Fig. 1. Market Pressure Effects on MinneTAC and Available Market Share

the customers. Both of these actions are equivalent to reducing competition and pressure in the marketplace.

The configurable Market Pressure Agent does the opposite: it increases available supply to customers, allowing them to pay less for computers while simultaneously putting more demand on suppliers, encouraging them to increase their prices. Our Market Pressure Agent operates by continually adjusting its customer offer prices to achieve a desired market share. In our experiments we set the market pressure agent to achieve a certain market share across all customer market sectors uniformly, but the agent could be configured to achieve any arbitrary level of the customer market in each sector independently. Since the agent is a build-to-order agent, it purchases parts to build the computers ordered, and the market share achieved on the customer side is reflected on the supplier side. This agent can capture market share on the interval 0% to 100%[3] because it has an unlimited line of credit and has no concern for its own profit-earning capability. When the Market Pressure Agent captures the desired customer market share, no other agent(s) can use that portion of the customer market: the Market Pressure Agent creates pressure in the marketplace.

By using combinations of Market Relief and Market Pressure Agents, developers can control the market environment. Figure 1 shows how the setting of Market Pressure affects remaining available market share as well as market share

[3] Note that in TAC-SCM, an agent can bid on and win a customer order without actually fulfilling the order. In this way, a single Market Pressure Agent is capable of absorbing 100% of the customer market share even though it could not actually fill these orders due to capacity constraints.

actually obtained by an agent under observation[4]. When a developer makes alterations to a competitive agent they wish to observe, they can use these tools in concert with the repeatable-game server to benchmark and compare the change in performance of their alteration.

4 Experiments

Using our Market Pressure Agent and a server set for locked-default values of the customer and supplier random variables[5], we examine what happens to an agent when we alter the levels of customer demand available in the marketplace. Figure 2, shows a composite response of the MinneTAC [8] agent over 12 locked-default games against a Market Pressure agent with various settings for customer market share absorbed. MinneTAC earns a relatively constant profit (the height of the middle band) until the Market Pressure Agent absorbs a very large portion of the market share[6] and begins to compete with MinneTAC for customers. At this point, MinneTAC loses market share because of the smaller available customer base. While this general behavior is to be expected in any marketplace, knowing the specific actions an agent will take in a competitive environment is very important when trying to diagnose if the agent is making correct decisions. We next examine several underlying indicators of specific behavior with respect to market pressure: revenue, profit, and cost.

In Figure 2 we depict the key profit components, measured at 12 different levels of available market customer orders. Here we notice some interesting trends: while the revenue is lower when there is less customer demand, our aggregate costs are also higher (because of competition with the Market Pressure Agent) which further exacerbates a decline in profit. When we look at the costs per-product in Figure 3 we can see exactly how much of our potential profit is consumed by these costs.

Developers can use a detailed performance analysis such as this to uncover undesirable behaviors in their agents. One behavior that we discovered while evaluating MinneTAC was that as customer demand decreased, the amount MinneTAC spent on storage fees[7] per order increased 20-fold, as shown in Figure 4. In fact, as a fraction of total costs, storage costs range from approximately 1.5%

[4] This graph was generated from a server using locked-default values for all random variables for the customers and suppliers.

[5] Recall that for this method, we set default values for all $[min, max]$ intervals that were configureable within the server. Thus our new settings for each interval are $[x, x]$ where $x = \frac{(min+max)}{2}$.

[6] We measure market share using the *CMieux Analysis and Instrumentation Toolkit for TAC SCM* software [9] developed at the Carnegie Melon University. This software is available at http://www.cs.cmu.edu/~mbenisch/ait/ or in the 3rd party software section of the SICS TAC-SCM website.

[7] In a standard TAC-SCM competition, the storage rates are randomly determined constants on the interval [25%, 50%] The storage rate that will be used for an entire game is selected at random and broadcast to all agents at the start of the game.

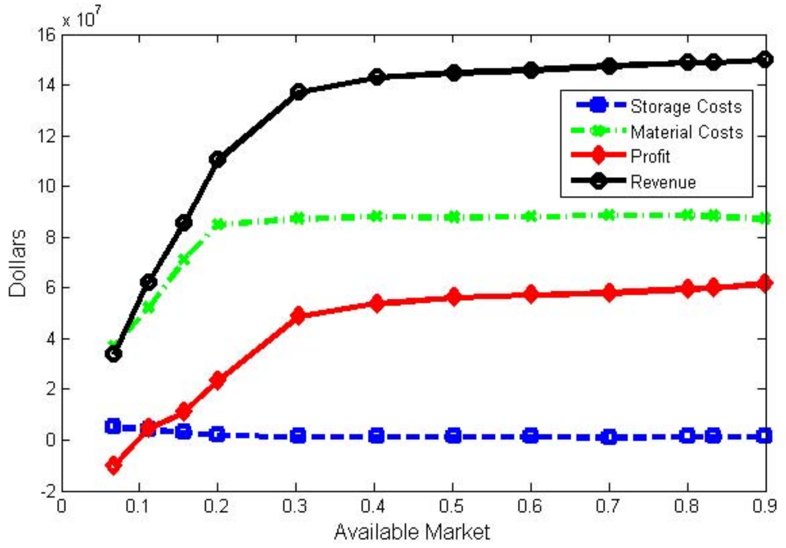


Fig. 2. Market Pressure Effects on MinneTAC Revenue

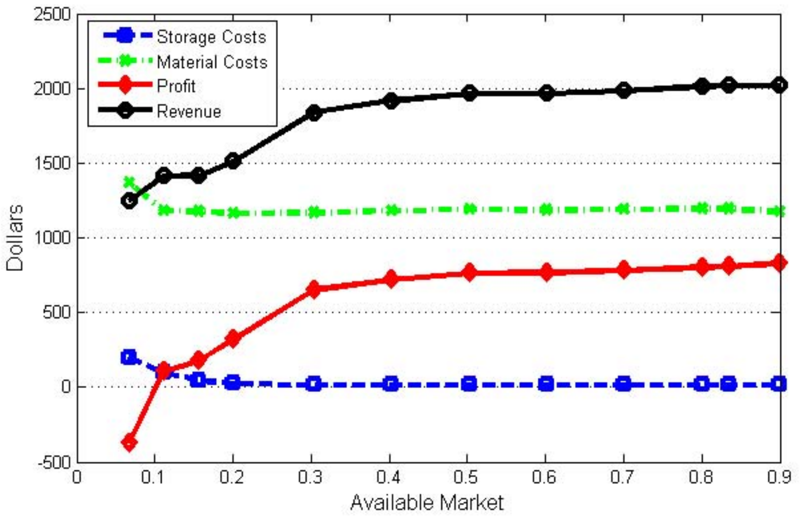


Fig. 3. Market Pressure Effects on MinneTAC Revenue (per product unit)

of the total costs under low pressure markets to over 12% of the total cost in high pressure markets. After investigating the problem, the code developers discovered that the agent was not considering storage costs in the profit function being optimized. If the profit function had included the storage costs in the calculation, it probably would have purchased and held fewer components or sold computers at a more competitive price to avoid excessive storage fees.

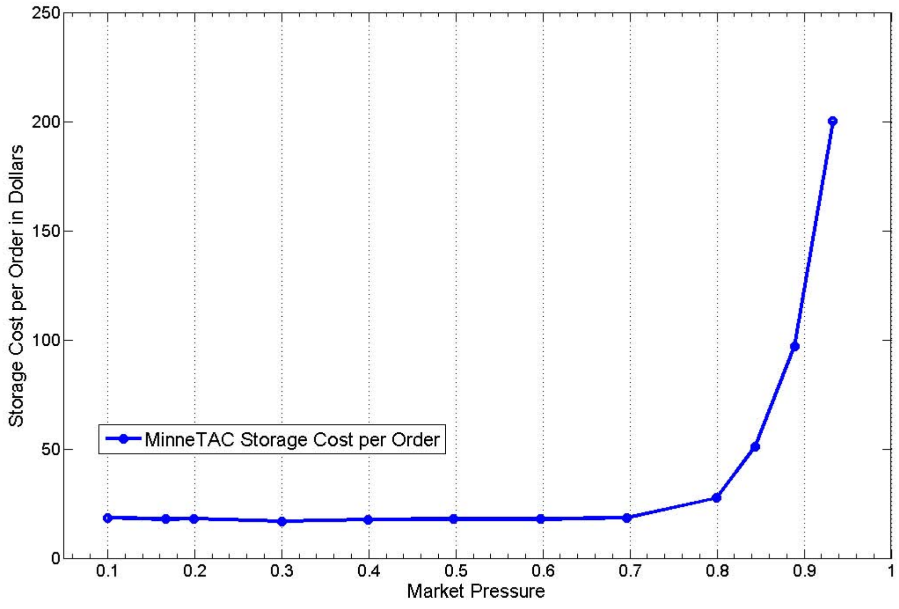


Fig. 4. Market Pressure Effects on MinneTAC Storage Costs

For our next experiment, we wanted to compare the performance of several agents under various market conditions: the winning agent in the TAC-SCM 2005 competition, TacTex (University of Texas at Austin); the 3rd place finalist, Mertacor (Aristotle University of Thessaloniki, Greece); and MinneTAC, our agent that took 5th place in the finals. Since we wanted to see how these agents would behave under various market conditions during games with unlocked random variables, we decided to replay the same full-random game for each agent, with the only variation being what competitor was selected, and what the pressure setting on the Market Pressure Agent was. We set the server to use our repeatable mode[8] and examined the performance of these agents individually under various market pressure conditions.

We used unmet demand (the amount of unfulfilled customer orders as a fraction of the total number of customer orders) as a measure of market pressure. The fewer the number of unmet customer orders, the more downward product-price pressure there is amongst competing manufacturers. Figure 5 shows the performance of the three agents at various levels of unmet customer orders. Notice that TacTex outperforms the other two agents by a wide margin ($20M in this game) until there are very few unmet customer orders remaining (approximately

[8] Recall that repeatable mode uses the modified server set with the identical initial random seeds for controlling the consumer and supplier random variables generated for every game. This allows independent control of the sequence of random numbers generated for the server processes, tie-breaking, and agent behavior.

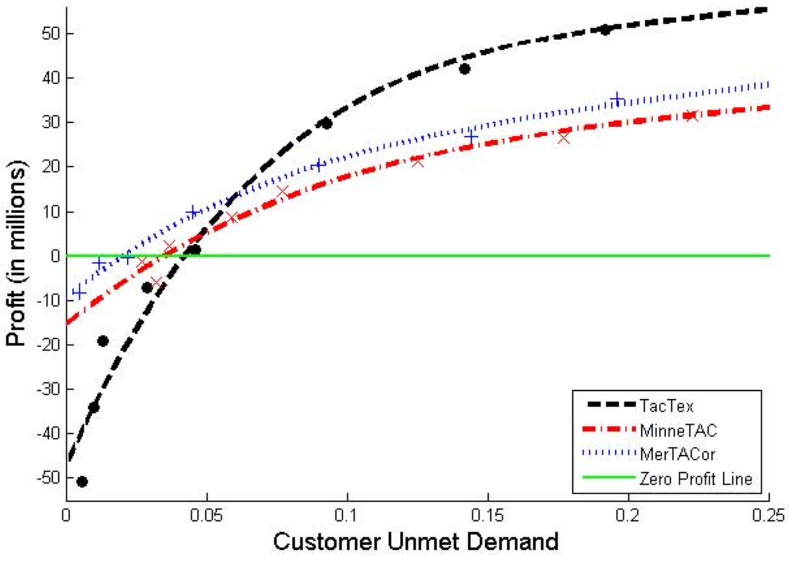


Fig. 5. Comparison of Agent Performance. In this figure, the closed circles represent actual values for TacTex, the x's represent actual values for MinneTAC and the +'s represent actual values for Mertacor. The lines represent the best-fit polynomial curves for each agent's data.

7% or about 1800 orders in this game). At this point, TacTex performance drops significantly, and the other two agents perform better. Interestingly, in the 2005 TAC-SCM finals games, the unmet customer demand varied from 7% to 27% across all games. As seen in the figure, in this region of unmet demand, TacTex has the best performance of the three agents we tested.

If we consider the area under each competitor's profit curve as another indication of agent performance, we can determine not only at which unmet demand levels does an agent do well, but also how well the agent will perform under a spectrum of market conditions. By making changes that increase the area under the profit curve, developers can improve their agent's performance against yet-to-be-seen competitors.

5 Conclusion

We developed a framework of tools to control the TAC-SCM simulation environment and measure the performance of trading agents under various market conditions. Since we are likely to see high market pressures exhibited by competitors in actual competitions, we can use these market pressure tools to simulate various market pressure levels and discover undesirable agent behavior before the agent is used in competition. Furthermore, teams can now evaluate their agents in a variety of market environments that were previously unable to be simulated with combinations of existing past competitors. Finally, these tools allow research teams to better understand how individual changes to their agents affect performance in various market conditions. We believe that agents optimized with the market pressure tools will perform better in actual competition.

6 Future Work

While we've provided a benchmark framework and a set of tools that will allow developers to characterize the performance of their agents against a standard reference, there are many areas in the framework that need refining.

For example, while our current Market Pressure Agent has complete control over how many bids it generates for customer RFQs, it has only a limited influence over the pressure it creates in the supply-side chain. While it is clear how to capture an arbitrary percentage of the customer market, it remains unclear what the corresponding meaning of market share is on the supplier side since supplier side contracts have an additional dimension (delivery time) that must be considered. When a new customer order arrives, the Market Pressure Agent purchases parts immediately in high quantities with an as-soon-as-possible delivery date. But because other agents compete for parts in 3 dimensions (quantity, price, and delivery date), and the Market Pressure Agent only operates in the quantity and price dimensions, the Market Pressure Agent is unable to create pressure against future purchases of parts the other agents make. As a result, our agent tightly controls market share in the product market, but it only loosely influences the parts market. Resolving this problem is a topic for future research.

A second area of future work is to develop a variable-pressure agent. Instead of trying to achieve a fixed pressure level for the entire game, this new agent could achieve different market pressures for pre-determined intervals throughout the game. We could then examine the ability of an observed agent to react to the change in market pressure to measure its level of adaptability.

References

1. Collins, J., Arunachalam, R., Sadeh, N., Ericsson, J., Finne, N., Janson, S.: The supply chain management game for the 2006 trading agent competition. Technical Report CMU-ISRI-05-132, Carnegie Mellon University, Pittsburgh, PA (2005)
2. Minghua He, Alex Rogers, D.E., Jennings, N.R.: Designing and evaluating an adaptive trading agent for supply chain management applications. In: IJCAI-05 Workshop on Trading Agent Design and Analysis TADA-05, Edinburgh, Scotland (2005)
3. Minghua He, Alex Rogers, X.L., Jennings, N.R.: Designing a successful trading agent for supply chain management. In: Fifth International Joint Conference on Autonomous Agents and Multiagent Systems, (Hakodate, Japan)
4. Kiekintveld, C., Vorobeychik, Y., Wellman, M.P.: An analysis of the 2004 supply chain management trading agent competition. In: IJCAI-05 Workshop on Trading Agent Design and Analysis TADA-05, Edinburgh, Scotland (2005)
5. Pardoe, D., Stone, P.: TacTex-03: A supply chain management agent. SIGecom Exchanges: Special Issue on Trading Agent Design and Analysis **4**(3) (2004) 19–28

6. Pardoe, D., Stone, P.: Bidding for customer orders in TAC SCM. In: AAMAS 2004 Workshop on Agent Mediated Electronic Commerce VI. (2004)
7. Pardoe, D., Stone, P.: Predictive planning for supply chain management. In: Proceedings of the International Conference on Automated Planning and Scheduling. (2006)
8. Ketter, W., Kryzhnyaya, E., Damer, S., McMillen, C., Agovic, A., Collins, J., Gini, M.: Analysis and design of supply-driven strategies in TAC-SCM. In: AAMAS-04 Workshop on Trading Agent Design and Analysis, New York (2004) 44–51
9. Benisch, M., Andrews, J., Bangerter, D., Kirchner, T., Tsai, B., Sadeh, N.: Cmieux analysis and instrumentation toolkit for tac scm. Technical Report CMU-ISRI-05-127, School of Computer Science, Carnegie Mellon University (2005)

Competing Sellers in Online Markets: Reserve Prices, Shill Bidding, and Auction Fees*

Enrico H. Gerding, Alex Rogers, Rajdeep K. Dash, and Nicholas R. Jennings

University of Southampton, Southampton, SO17 1BJ, UK
Tel.: +44 (0) 23 8059 9201, Fax: +44 (0) 23 8059 2865
{eg,acr,rkd,nrj}@ecs.soton.ac.uk

Abstract. In this paper, we consider competition between sellers offering similar items in concurrent online auctions, where each seller must set its individual auction parameters (such as the reserve price) in such a way as to attract buyers. We show that in the case of two sellers with asymmetric production costs, there exists a pure Nash equilibrium in which both sellers set reserve prices above their production costs. In addition, we show that, rather than setting a reserve price, a seller can further improve its utility by shill bidding (i.e., pretending to be a buyer in order to bid in its own auction). But, through the use of an evolutionary simulation, we show that this shill bidding introduces inefficiences within the market. However, we then go on to show that these inefficiences can be reduced when the mediating auction institution uses appropriate auction fees that deter sellers from submitting shill bids. Specifically, we compare two types of auction fees and show that, in this respect, those based on the difference between the closing price and the reserve price are more effective than the commonly used fees that are based on closing price alone.

1 Introduction

Online markets are becoming increasingly prevalent and extend to a wide variety of areas such as e-commerce, Grid computing, recommender systems, and sensor networks [2,5,12]. To date, much of the existing research has focused on the design and operation of individual auctions or exchanges for allocating goods and services. In practice, however, similar items are typically offered by multiple independent sellers that compete for buyers and set their own terms and conditions (such as their reserve price and the type and duration of the auction) within an institution that mediates between buyers and sellers. Examples of such institutions include eBay, Amazon and Yahoo!, where at any point in time multiple concurrent auctions with different settings are selling similar objects, often resulting in strong competition[1]. Given this competition, a key research question is how

* This research was undertaken as part of the EPSRC funded project on Market-Based Control (GR/T10664/01).

[1] To illustrate the scale of this competition, within eBay alone close to a thousand auctions for selling Apple's iPod nano were running worldwide at the time of writing.

a seller should select their auction settings in order to best attract buyers and so increase their expected profits. In this paper, we consider this issue in terms of setting the seller's reserve price (since the role of the reserve price has received attention in both competitive and non-competitive settings). In particular, we extend the existing analysis by considering how sellers may improve their profit by shill bidding (i.e. bidding within their own auction as a means of setting an implicit reserve price), and moreover, we investigate how the institution can deter this undesirable shill bidding through the use of appropriate auction fees.

In more detail, the existing literature on competing sellers (see section 5) has shown that setting a reserve price has two opposing effects on the seller's profit. On the one hand, it increases the expected profit by guaranteeing a minimum price in the case that the item is sold. On the other hand, it deters potential buyers from participating, and thus decreases the expected profits. However, this analysis neglects the possibility that sellers may avoid the latter by shill bidding (i.e., by covertly bidding within their own auction) and announcing a low reserve price, thereby overcoming the disadvantage of deterring bidders and, at the same time, ensuring that the item is not sold at too low a price. However, shill bidding undermines buyers' trust and has an adverse effect on market efficiency, since setting a reserve price allows the bidders to make informed decisions of which auction to attend. For these reasons, shill bidding is illegal in many countries. Nevertheless it is one of the most common forms of internet auction fraud [8,13], and is often hard to detect in online auctions, where participants are relatively anonymous. To this end, in this paper, in addition to the above issues, we investigate how auction fees (i.e. the payments made by the seller to the institution for its services as a mediator) can be used by an institution to reduce the incentive for shill bidding within a setting of competing sellers. More specifically, we make the following contributions:

- We analytically describe the seller's equilibrium strategies for setting reserve prices for the two-seller case, and we advance the current state-of-the-art by finding Nash equilibria by iteratively discretising the search space. We show that, although no pure strategies exist when the sellers are symmetric, these can be found if production costs differ sufficiently between the two sellers.
- For the first time, we investigate shill bidding within a setting of competing sellers. To this end, we derive analytical expressions for the seller's expected utility when sellers shill bid. Using these expressions, we show that, without auction fees, a seller can considerably benefit by shill bidding when faced with competition.
- We compare various types of auction fees, and evaluate their ability to deter shill bidding and their impact on market efficiency. Since the efficiency and the sellers' equilibrium strategies in case of auction fees cannot be calculated analytically, we introduce an evolutionary simulation that allows us to simulate the market environment, and learn the seller's strategies when auction fees are applied. Using this simulation, we show that, by setting appropriate auction fees, an institution can both deter shill bidding and increase efficiency. This analysis is novel for a market with competing sellers.

The remainder of the paper is organised as follows. Section 2 describes the model of competing markets in detail. Section 3 analyses the buyer and seller strategies, and identifies the cases for which a pure Nash equilibrium exists. Section 4 compares the analytical results to the outcomes using the evolutionary simulation, and extends the results by introducing auction fees and measuring market efficiency. An overview of the related work is given in section 5 and section 6 concludes.

2 Model of Competing Sellers

The model of competing markets consists of buyers, sellers, and a mediator. Each seller is associated with a separate auction. The mediator is an institution such as eBay or Yahoo! that runs the actual auctions and acts as a broker between the buyers and the sellers. The competing sellers game consists of four stages and proceeds as follows (details are given below). In the first stage, the mediator announces the auction fees to the sellers. After observing the fees, the sellers simultaneously post their reserve prices in the second stage. In the third stage, the buyers simultaneously select an auction (or, equivalently, a seller) based on the observed reserve prices. In the final stage, the buyers submit their bids and the auctions are executed concurrently. In addition, a seller can also participate in the final stage by placing a shill bid.

In line with existing research on competing markets, we assume each bidder selects at most one auction [3,7,10]. This assumption is reasonable when items are complete substitutes and a buyer requires only one item. Although the model can be extended to allow for complementarities and participation in multiple markets, this introduces additional challenges for the analysis of bidder and seller strategies, and thus, here, we seek to understand the canonical behaviour.

2.1 The Mediator

The mediator decides on the auction fees and determines the market rules or *mechanism* to be used in the auctions. In our current model, we use a second-price sealed bid (or Vickrey) auction, in which the highest bidder wins but pays the price of the second-highest bidder. This mechanism was chosen because: (i) it requires little communication and is fast to execute, and (ii) it requires minimal reasoning on the part of the buyers since bidding the true value is a weakly dominant strategy [9]. Moreover, previous research has shown that, for the single-unit auction setting described above, the second-price auction is an equilibrium strategy in the competing sellers game in which a seller can select any direct mechanism [10]. Although many other equilibria exist, the expected revenue for the buyer is equivalent and thus does not affect the buyer's decision of seller. Therefore, the results presented here are not limited to the second-price auction, but generalise to any other mechanism that allocates the good to the buyer with the highest valuation (provided this valuation exceeds the reserve price), such as the English and the first-price auction.

In this work, we consider the following two types of commission fees:[2]

- **Closing Price (CP).** This fee is paid by the seller only if the item is sold, and is a fraction β of the closing price, where β is the CP commission rate.
- **Reserve-Difference (RD).** This variant of the first fee is calculated as a fraction δ of the difference between the selling price and the seller's declared reserve price, where δ is the RD commission rate. Note that this fee is essentially a combination of a CP fee and a negative RD fee.

The first type of fee is the most common in online auctions such as eBay, Yahoo! and Amazon. In practice the rate is often not a constant, but depends on factors such as the domain (books, cd's, computers, etc.) and the price range. However, since here we investigate a general model and abstract away from any specific domain, we assume a constant rate. The second type of fee is introduced in previous literature, where it is called the commision fee and is shown to prevent shilling for particular bidder valuation distributions in a single multi-stage auction [13]. Similarly, our aim is to apply auction fees in order to reduce the incentive of a seller to shill bid. However, in addition, we are concerned with how efficiency can be improved using such fees. To this end, we compare the effectiveness of this fee with the more established CP commission fee.

2.2 The Sellers

A seller has the option to openly declare a minimum or *reserve* price. In addition, the seller is able to *shill bid*. If the shill bid wins the auction, effectively no sale is made. However, a seller is still required to pay the auction fees.

2.3 The Buyers

A buyer first selects a single auction based on the announced reserve price, and then bids in the selected auction. The bidding is not affected by the reserve price; it is a weakly dominant strategy to bid the true value [9]. On the other hand, the reserve price is an important factor in determining which auction the buyer should choose. To this end, the buyer's equilibrium strategies for selecting an auction are detailed in the next section.

3 Analysis

We now analyse the bidder and seller strategies for the above model. A complete analysis of the players' equilibrium behaviour and of the market efficiency for the

[2] Other common types of fees in online auctions are the *Listing Fee* and the *Reserve Price Fee*. The former is an upfront fixed payment for listing the item. Because the fee is payed irrespective of the outcome or the reserve price, it does not affect a seller's behaviour, and so we do not consider it here [13]. The latter is a fraction of the declared reserve price, and is also paid upfront. We do not consider this fee because, when it is positive, it has an adverse effect and actually provides an additional incentive for the seller to shill (this is then called *reserve price shilling* [8]).

above model is intractable [10]. Therefore, in this section, we analyse a simplified version with two sellers and without auction fees (the complete model with auction fees and the market efficiency are then investigated using a simulation approach in section 4). Here, we extend the analysis in [3] by locating pure seller Nash equilibria for cases in which there is and there is not shill bidding. Following [3,7,10], we assume that each buyer only requires one item, each seller offers one item for sale, and all items are perfect substitutes (i.e., the buyer's choice of seller is based on the declared reserve prices only). As mentioned earlier, each buyer can attend at most one auction. Furthermore, all buyer and seller preferences are described by von Neumann and Morgenstern utility functions, and players are assumed to be risk neutral.

We use the following notation. Seller i's reserve price and shill bid are denoted by r_i and s_i resp. Without loss of generality, we assume $r_j \geq r_i$ if $j > i$. Each seller has production costs x_i. A buyer's valuation is denoted by v, and N is the total number of buyers. The buyer valuations are independently drawn from the set $[0, 1]$ according to a commonly known cumulative distribution F with a density f and support $[0, 1]$.

3.1 Buyer Equilibrium Behaviour

The buyer policy for two sellers has been analysed in [3], and has been extended for multiple sellers in [7]. Clearly, a rational buyer with valuation $v < r_1$ will not attend any auction. Furthermore, if $r_1 < v < r2$, the buyer will always go to seller 1. The interesting case occurs when $v > r2$. In a symmetric Nash equilibrium, there is a unique cut-off point $1 \geq w \geq r2$ where buyers with $v < w$ will always go to seller 1, and buyers with $v \geq w$ will randomize between the two auctions with equal probability. The cut-off point w is exactly where a buyer's expected utility is equal for both auctions, and is thus found by solving the following equation (for proof see [3]):

$$r_1 \mathcal{F}(r_1, w)^{N-1} + (N-1) \int_{r_1}^{w} y \mathcal{F}(y, w)^{N-2} dF(y) = r_2 \mathcal{F}(w, w)^{N-1} \qquad (1)$$

where $\mathcal{F}(y, w) = F(y) + [1 - F(w)]/2$. When r_2 is close to 1, for some values of r_1 equation 1 never holds, and thus, $w = 1$ (i.e., auction 1 is always better, and thus all buyers go to this auction). Given the buyers' cut-off point, we can now calculate the expected revenue for the sellers.

3.2 Seller Equilibrium Behaviour

In order to calculate the equilibrium behaviour of the sellers, we first derive a general expression for the sellers' expected utility. We then apply this result to three different cases: (i) where both sellers declare public reserve prices, (ii) where one seller declares a public reserve price and the other submits a shill bid, and (iii) where both sellers submit shill bids. When a seller decides to shill bid, the declared reserve price does not offer any additional benefit. We therefore assume in the following that a seller that shills declares no reserve price (or, equivalently, a zero reserve price).

Sellers' Expected Utility. The utility of a seller in any auction is calculated by considering the probability of one of three events occurring: (i) no bidders having valuations above the reserve price and the item does not sell, (ii) only one bidder having a valuation above the reserve price and the item sells at the reserve price, or (iii) two or more bidders having valuations above the reserve price and the item sells at a price equal to the second highest valuation. Thus, the expected utility of seller i, assuming that they have a production cost of x_i and set a reserve price of r_i is given by:

$$U_i(r_i, x_i) = N(r_i - x_i)\mathcal{G}(r_i)(1 - \mathcal{G}(r_i))^{N-1} \\ + N(N-1)\int_{r_1}^{1}(x_i - y)\mathcal{G}'(y)\mathcal{G}(y)(1-\mathcal{G}(y))^{N-2}dy \quad (2)$$

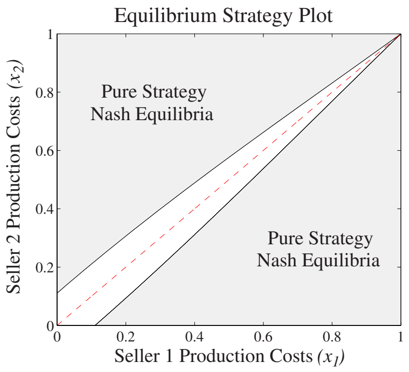
where $\mathcal{G}(y)$ denotes the probability that any bidder is present in the auction *and* that this bidder has a valuation greater than y.

Now, in the standard auction with no competing sellers, we have the standard result that $\mathcal{G}(y) = 1 - F(y)$ and $\mathcal{G}'(y) = -f(y)$ (see [9], Ch. 2). However, in the case of two competing sellers, we must modify this expression for $\mathcal{G}(y)$ to account for the fact that the number and valuation of the bidders that attend each of the auctions is determined by the bidders' cut-off point w. Thus, for sellers 1 and 2 (where seller 1 has the lower reserve price), $\mathcal{G}_1$ and $\mathcal{G}_2$ are given by:

$$\mathcal{G}_1(y) = \begin{cases} \frac{1+F(w)}{2} - F(y) & y < w \\ \frac{1-F(y)}{2} & y \geq w \end{cases} \qquad \mathcal{G}_2(y) = \begin{cases} \frac{1-F(w)}{2} & y < w \\ \frac{1-F(y)}{2} & y \geq w \end{cases} \quad (3)$$

These expressions show that the expected utility of each seller is determined, not just by the reserve price that they themselves set, but also by the reserve price set by the competing seller (since, both these values determine the bidders' cut-off point, w, and thus the number and valuation of bidders who attend each auction). Thus the equilibrium behaviour of the sellers is complex, and we consider each of the three cases discussed earlier separately.

Both Sellers Announce Public Reserve Prices. When both sellers announce public reserve prices, the equilibrium strategy of each seller will be given by a Nash equilibrium, at which each seller's reserve price is a utility maximising best response to the reserve price of the competing seller. In the case of symmetrical sellers, where $x_1 = x_2$, it is known that no pure strategy Nash equilibrium exists. Since each seller can maximise their expected utility by setting a reserve price marginally below that of the competing seller, a mixed strategy equilibrium results [3]. However, where the sellers have production costs that are sufficiently different from one another, this is not the case. Whilst undercutting the competing seller still yields some benefit, it is more advantageous to announce a higher reserve price that attracts less bidders but generates greater revenue. Thus, a pure Nash equilibrium exists where the reserve price of both sellers is higher than their production costs.



(a)

		Seller 2	
		Reserve Price	Shill Bid
Seller 1	Reserve Price	0.452 , 0.189	0.403 , 0.220
	Shill Bid	0.457 , 0.188	0.423 , 0.220

(b)

Fig. 1. (a) The grey area shows the combinations of production costs for which a pure Nash equilibrium exists ($N = 10$ and buyer valuations are distributed uniformly). **(b)** Example demonstrating the effect of shill bidding on the expected utility of sellers 1 and 2 ($N = 10$, $x_1 = 0.25$, and $x_2 = 0.5$).

Although we find a complete characterisation of this equilibrium solution intractable, we are able to find pure strategy Nash equilibria for specific cases by iteratively discretising the space of possible reserve prices. That is, for all possible values of r_1 and r_2 that satisfy the conditions $x_1 \leq r_1 \leq 1$ and $r1 \leq r_2 \leq 1$, we calculate w and hence the expected utility of the two sellers. We then search these reserve price combinations to find the values of r_1^* and r_2^* that represents the utility maximising best responses to one another. By iterating the process and using a finer discretisation at each stage, we are able to calculate the Nash equilibrium to any degree of precision, and we can confirm that this is indeed the pure Nash equilibrium by checking that the utility of seller 2 cannot be further improved by undercutting seller 1 (i.e. $U_2(r_2, x_2) < U_2(r_2^*, x_2) \; \forall \; r_2 < r_1^*$). Figure 1a shows a plot indicating the range of asymmetric cases (i.e., cases where $x_1 \neq x_2$) in which we find a pure strategy Nash equilibrium. As can be seen, the symmetric case is very much a special case, and the majority of possible production cost combinations yield unique pure strategy Nash equilibria, at which we can calculate the seller's expected utility.

One Seller Shill Bids. Rather than announce a public reserve price, either of the sellers may choose to announce a reserve price of zero to attract bidders, and then submit a shill bid to prevent the item from selling at too low a price. Thus at this equilibrium, we must determine the value of the shill bid submitted and the value of the reserve price set by the seller who does not shill bid. However, since the value of the shill bid does not affect the revenue of the other seller, we are not looking for a Nash equilibrium, but can consider each seller individually.

Thus, the seller who does not shill bid (seller 2 since r_2 will be greater than r_1) should declare a reserve price that is a best response to the zero reserve price announced by the bidder who does shill bid. This reserve price is simply given by the value of r_2 that maximises $U_2(r_2, x_2)$, given that we calculate $\mathcal{G}_2(y)$ as in equation 3 and taking $r_1 = 0$ in order to calculate w.

Given the best response reserve price of seller 2, and the resulting value of w, we can also calculate the shill bid that seller 1 should submit in order to maximise its own expected utility. By substituting s_1 for r_1 in equation 2, and using $\mathcal{G}_1(y)$ as given in equation 3, we find the shill bid that maximises $U_1(s_1, x_1)$.

Both Sellers Shill Bid. Finally, we consider the case that both sellers choose to declare a zero reserve price and both submit shill bids. In this case, the bidders will randomise equally between attending either auction, since there is no reserve price information to guide their decision. Thus we can find the equilibrium shill bids of both sellers, by again substituting s_i for r_i in equation 2 and hence finding the value of s_i that maximises $U_i(s_i, x_i)$ when $w = 0$.

Figure 1b shows a typical example of the four resulting strategy combinations displayed as a normal form game. Note that in this example, both sellers have a dominant strategy to submit shill bids, and thus this result is a Nash equilibrium (this result holds in general in the absence of auction fees). At this equilibrium seller 2 achieves its maximum possible utility. However, seller 1 receives less utility at this equilibrium than in the case when neither seller shill bids. Thus, seller 1 is better off with a mechanism that deters all parties from submitting shill bids, and we now consider how this can be achieved through the use of appropriate auction fees.

4 Auction Fees and Market Efficiency

So far, we have ignored auction fees and market efficiency in the analysis of the competing sellers game. In this section we compare the different types of auction fees described in section 2.1, and consider which is most effective at deterring shill bidding. Furthermore, we investigate to what extent the market is efficient and how auction fees and shill bidding by the seller affect this efficiency. As discussed previously, efficiency is a desirable property since an efficient market extracts the maximum surplus that is available. It is therefore important to take efficiency into consideration when finding appropriate auction fees.

However, the presence of auction fees adds considerable complexity to the analysis of the competing sellers game. Without them, we can reasonably assume that a seller declares a zero reserve price in combination with a shill bid. With them, on the other hand, this is not necessarily a utility-maximising strategy (i.e., a seller may benefit by having both a shill bid and setting a non-zero reserve price). Thus, the dimensionality of the strategy space is significantly increased.

In view of the above, we use a simulation based on evolutionary algorithms (EAs) to investigate the competing sellers game with auction fees and the efficiency of the market. EAs are chosen because they provide a powerful metaphor for learning in economics. In addition, they have been successfully applied in the past to settings where game-theoretic solutions are not available [1,2,6].

In the following, we first describe the evolutionary simulation (section 4.1). We then evaluate the simulation by using analytical outcomes as a benchmark in section 4.2. In section 4.3 we investigate the efficiency of the competing sellers game with and without shill bidding. Lastly, we show the results using auction fees and how these affect shill bidding and efficiency.

4.1 Evolutionary Simulation

The simulation evolves a population of seller strategies using an EA. Here, a seller strategy consists of both the declared reserve price and a shill bid, and is encoded on the so-called chromosome using real values.[3] We allow each seller to be of a different *type*, where this type is determined by its production costs. In case the types differ (i.e., the sellers are asymmetric), the chromosome contains a separate reserve price and shill bid for each type. The *fitness* or performance of each strategy is evaluated by executing the competing sellers game with M sellers as described in section 2, and works as follows. First, M seller individuals are randomly selected from the population and compete in a number of consecutive games. In these games, each seller sets the reserve price and shill bid according to its strategy. The bidder strategies are calculated numerically by solving equation 1. The fitness values of the individuals are then calculated by taking the average obtained utility by each seller. In case a strategy represents several types of players, the strategies are evaluated several times with different roles for the seller in each evaluation. In other words, the sellers are assigned a different type in each evaluation to obtain a good estimate of the performance of the entire chromosome. The process of selecting and evaluating individuals is repeated until all the individuals of the population are evaluated.

The fittest seller individuals survive and are transferred to the next generation, whereas poor performing individuals are removed from the population. New strategies are also explored by slightly modifying existing individuals using a mutation operator. This evolutionary process is repeated for a fixed number of iterations.

Note that the EA is not simply used to find an optimal solution for a static problem, as the optimal strategy depends on the strategies of other individuals in the population, resulting in complex dynamics. Furthermore, although we apply the simulation to the case of two sellers in this paper (thus enabling us to compare simulation results with the analytical results derived in section 3), it is in no way restricted to this case. Increasing the number of sellers does not require any change in methodology, although it does increase the complexity of the search task.

The general settings used for all experiments reported in this paper are as follows: the number of sellers $M = 2$, the population size is 30, and the evolutionary results are obtained after 1000 generations. Each strategy is evaluated by playing 1000 competing sellers games with randomly generated buyers. The buyer valuations are selected from a uniform distribution with support $[0, 1]$. In order to obtain statistically significant results, we report average results and

[3] A class of EAs called *Evolution Strategies* are most appropriate where strategies are encoded using real values, and thus we use them here. For more details, see [6], where a very similar methodology was used for bilateral negotiation.

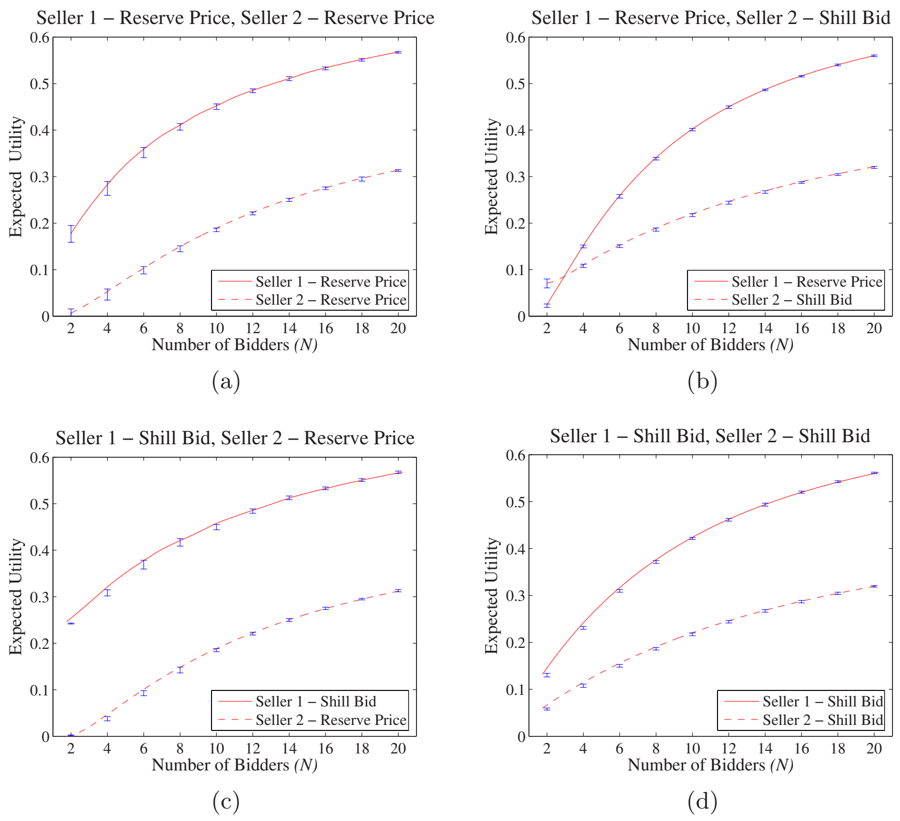


Fig. 2. Plots showing agreement between analytical Nash equilibrium (shown as continuous curves) and evolutionary results (shown as error-bars representing the standard deviation) for the competing sellers game with varying number of buyers for the cases where no seller shill bids (a), one of the sellers shill bids (b and c), and both sellers shill bid (d). Production costs are set to $x_1 = 0.25$ and $x_2 = 0.50$.

standard deviations of 30 runs of the simulation using the same settings but with different random seeds.

4.2 Evolutionary vs. Analytical Results

We first apply the analytical results from section 3.2 as a benchmark to evaluate the evolutionary simulation. To this end, figure 2 shows both the Nash equilibrium solutions (calculated as described in section 3.2) and the evolutionary results for cases with and without shill bidding. In these experiments, seller 1 and 2's production costs are set to 0.25 and 0.5 respectively. These settings were chosen to illustrate representative outcomes when both sellers have non-zero and asymmetric production costs. As can be seen from figure 2, the results show a perfect match. We also experimented with other combinations of production costs in the range between 0 and 0.5, and for cases where both sellers shill bid and no seller shill bids, resulting in an equally good match between game-theoretic and evolutionary outcomes (results are not shown due to space limitations).

4.3 Efficiency

Efficiency is generally a desirable property of a market since it extracts the maximum surplus available. It is therefore important to investigate how shill bidding and auction fees affect efficiency. Here, we consider *allocative efficiency*, which is achieved when the item is awarded to the buyer with the highest valuation, or to no buyer if the highest valuation is below the production costs. More formally, we define allocative efficiency as follows:

Definition 1. Efficient Allocation. An efficient allocation is given by:

$$K^* = \arg\max_{K \in \mathcal{K}} \left(\sum_{i=1}^{N} v_i(K) - \sum_{i=1}^{M} x_i(K) \right),$$

where $\mathcal{K}$ denotes all possible allocations, $v_i(K)$ denotes bidder i's utility for a given allocation K, and $x_i(K)$ seller i's production costs for a given allocation. We measure the relative efficiency η of a given allocation $\hat{K}$ as follows:[4]

$$\eta = \frac{\sum_{i=1}^{N} v_i(\hat{K}) + \sum_{i=1}^{M} (x_i - x_i(\hat{K}))}{\sum_{i=1}^{N} v_i(K^*) + \sum_{i=1}^{M} (x_i - x_i(K^*))} \tag{4}$$

Now, a certain amount of inefficiency is inherent to the competing sellers game as a result of the buyers randomising over sellers. For example, if two buyers with the highest valuation both happen to choose seller 1, only one of them is allocated the item and allocative efficiency is not reached. In addition, shill bidding can lower the efficiency. This is in the first place because shill bidding enables a seller to hide production costs and therefore attract buyers that have no chance of winning. A second source of inefficiency arises because a declared reserve price is usually low due to competition. An optimal shill bid, on the other hand, is higher than a declared reserve price (and higher than production costs), resulting in less sales, and therefore a lower efficiency.

We are able to quantify this degree of inefficiency using the evolutionary simulation. In detail, for $N = 10$, $x_1 = 0.25$, and $x_2 = 0.5$, the average relative efficiency η when the sellers neither shill bid nor declare a reserve price is 0.948 ± 0.000, and is 0.946 ± 0.002 with reserve prices (in equilibrium). Clearly, the efficiency in these two cases is very similar. This is because the reserve prices in equilibrium are relatively low due to competition. With shill bidding, on the other hand, η is much lower and equals 0.910 ± 0.004 on average for these settings. Qualitatively similar results are obtained in other settings.

[4] Note that in order to prevent a negative value for η, we add production costs x_i in both the denominator and the numerator.

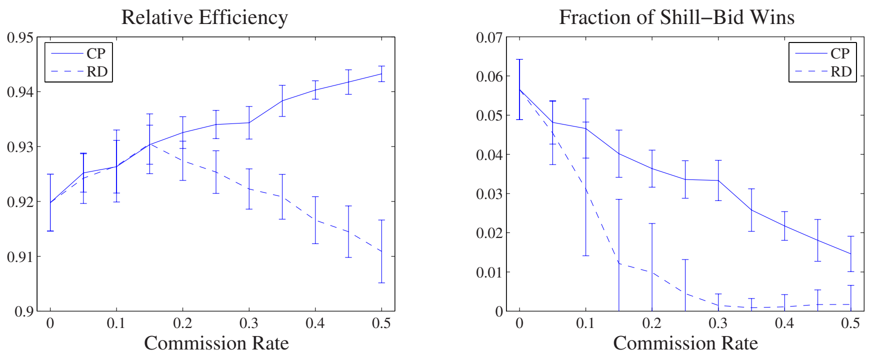


Fig. 3. Average relative efficiency η and fraction of shill-bid wins for two types of auction fees and various levels of commission rates, for $N = 10$ and $x_1 = x_2 = 0$. The error-bars indicate the standard deviation.

4.4 Auction Fees

In order to reduce the incentive for sellers to shill bid, and to increase the relative efficiency η, we introduce auction fees as described in section 2.1. Specifically, using the evolutionary simulation, we compare the effectiveness of the closing price (CP) commission fee and the reserve-difference (RD) commission fee (see section 2.1). Here, the effectiveness of a fee is measured by the relative efficiency η, as defined in section 4.3, and by the fraction of *shill-bid wins* (i.e., the fraction of auctions that result in a shill bid being the highest bid).

Thus, in order to directly compare the auction fees, we first consider the case without production costs. To this end, figure 3 compares the relative efficiency and fraction of shill-bid wins for the two types of auction fees. These results show that the RD fee is better able to reduce shill bidding for this setting. However, in case of high commission rates, a higher efficiency is obtained using the CP fee. This is because, as the RD commission fee is effectively a combination of a CP fee and a negative RD fee (see section 2.1), it rewards a high reserve price. If the commission rate is sufficiently high, the reserve price is set higher compared to the equilibrium price in absence of any auction fees, resulting in relatively inefficient outcomes. The CP commission fee, on the other hand, is neutral with regard to the reserve price.

In case of production costs we find similar results. However, if both production costs and CP commission rates are very high (e.g. both above 30%), a sharp decline in efficiency is observed using the CP fee. A seller will then set a very high reserve price to prevent making a loss in case it sells, and, as a result, efficiency becomes very low. Using the RD commission fee, on the other hand, efficiency does not decrease because sellers never make a loss as long as they report a reserve price above their production costs. The latter type of fee is therefore more robust to high production costs.

Another advantage of the RD auction fee is that sellers pay less fees on average, while obtaining the same efficiency gain. This is relevant because, if fees become too high, sellers will go to another mediator to sell their goods. Thus, an important goal of the mediator is to deter shill bidding and, at the same time, maintain reasonable rates. This goal is best achieved by the RD auction fee.

To conclude, the experiments show that in most cases highest efficiency is obtained by the commonly applied CP auction fees. The RD auction scheme, however, is more effective in deterring shill bidding and is also more robust to high production costs. This is consistent with earlier results showing that RD auction fees can deter shill bidding for isolated auctions [13]. However, our results show, for the first time, that these fees are also effective for a setting where sellers compete and in case of production costs. Moreover, we see that, when using the RD fee, sellers pay much less to the mediator overall compared to CP auction fees. The latter is especially important in a larger setting where multiple mediating institutions compete to attract sellers.

5 Related Work

McAfee [10] was the first to deal with mechanism design and reserve prices in the context of competing sellers. In his paper, sellers can choose any direct mechanism and these mechanisms are conducted for multiple periods with discounted payoffs for future periods. However, he assumed that (i) a seller ignores his influence on the profits offered to buyers by other sellers, and (ii) that expected profits associated with future profits are invariant to deviation of a seller in the current period. As McAfee notes, these assumptions are only reasonable in case of infinitely many players. This contrasts with our work, where we focus on the more realistic finite case with small numbers of buyers and sellers, where strategic considerations become important. Subsequent papers have relaxed some of the strong assumptions. In [3], a unique equilibrium strategy for the buyers in the two-seller case is derived (see also section 3.1). In addition, they show that in the seller's game there always exists an equilibrium, but it cannot be a symmetric one in pure strategies. They are unable to fully characterize the mixed equilibrium, but argue that sellers set a reserve price above their own valuation of the item. This analysis is generalised for a large number of sellers in [7], where it is shown that reserve prices tend to production costs in the limit case for asymmetric sellers. In this paper we extend the models in the above papers in two ways. Firstly, we introduce a mediator that charges commission fees to the seller for running the auction, and, secondly, we allow sellers to shill bid. Our extended model better reflects online markets, where commission fees and shill bidding are a common occurrence. Moreover, we are able to locate pure Nash equilibria for the asymmetric seller setting, and, for the first time, we measure the efficiency of the market with competing sellers using a simulation approach.

Our work is also closely related to recent research on simultaneous auctions. In [1,4] four types of auctions are considered: first-price sealed bid, second-price sealed bid, English, and Dutch auction. Due to the complexity of their framework, however, the proposed bidding strategies are based on heuristic approximations. In this paper, in contrast, results are grounded in a game-theoretic analysis with which we are able to calculate optimal seller reserve price settings. Furthermore, we argue that sellers compete with one another and need to adjust their reserve prices in order to attract bidders. This issue is not considered in [1,4], where auction parameter settings are a given.

Another related area is concerned with shill bidding. So far, shill bidding has been researched within isolated multi-stage auctions (such as the English auction [13,8]). Our paper, however, is the first one that considers shill bidding as a result of sellers having to compete. Moreover, we investigate what type of auction fees can best be used in order to reduce a seller's incentive to shill bid. This is important because shill bidding is often hard to detect. Apart from our work, the only paper that deals with preventing shill bidding is [13], where the RD fee is used. Although their paper indeed shows that this type of fee can prevent shilling, it does not consider competing sellers, production costs, and the efficiency of the market. By contrast, here we investigate these issues in detail and also compare the RD fee with the more established CP fee.

6 Conclusions

Traditionally, competition among sellers has been ignored when designing auctions and setting auction parameters. However, in this paper, we have shown that auction parameters (such as a reserve price) play an important role in determining the number and type of buyers that are attracted to an auction when faced with competition. We have also shown that such competition provides an incentive for sellers to shill bid, but this can be avoided by a mediator that applies appropriate auction fees. Now, these results become particularly relevant for online markets and multi-agent systems in which competition is rapidly growing due to the ease with which a buyer and/or a software agent can search for particular goods. Thus, in these settings, our results can be used by sellers seeking to maximise their profit, or alternatively, by the auction institution itself, who wishes to use appropriate auction fees to deter shill bidding and thus increase the efficiency of the market as a whole.

Research on competing sellers is a relatively young field and there are still a large number of challenges remaining. In future work, we intend to extend the analysis and the simulation results for shill bidding and auction fees beyond the case of two sellers. Since competition increases with more sellers, the incentive for sellers to shill bid and maintain a low reserve price becomes even stronger. As a result, we expect that setting appropriate auction fees to deter shill bidding will then become increasingly important. In addition, we intend to investigate the case where buyers require multiple items and can participate in multiple concurrent auctions. Ultimately, we would like to extend the concept of competition to the institutions themselves, and consider a model in which the actual institutions attempt to attract both sellers and buyers, whilst seeking to maximise revenue through their auction fees.

References

1. P. Anthony and N.R. Jennings. Developing a bidding agent for multiple heterogeneous auctions. *ACM Trans. on Internet Technology*, 3(3):185–217, 2003.
2. S. Bohte, E. Gerding, and H. La Poutré. Market-based recommendation: Agents that compete for consumer attention. *ACM Trans. on Internet Technology*, 4: 420–448, 2004.
3. R. Burguet and J. Sákovics. Imperfect competition in auction design. *International Economic Review*, 40(1):231–247, 1999.
4. A. Byde, C. Preist, and N.R. Jennings. Decision procedures for multiple auctions. In *Proc. 1st Int. Joint Conf. on Autonomous Agents and Multi-Agent Systems, Bologna, Italy*, pages 613–620. ACM Press, 2002.
5. S.H. Clearwater, editor. *Market-Based Control: A Paradigm for Distributed Resource Allocation.* World Scientific Publishing, 1996.
6. E.H. Gerding and H. La Poutré. Bilateral bargaining with multiple opportunities: Knowing your opponent's bargaining position. *IEEE Transactions on System, Man, and Cybernetics: Part C*, to appear.
7. Á. Hernando-Veciana. Competition among auctioneers in large markets. *Journal of Economic Theory*, 121:107–127, 2005.
8. R.J. Kauffman and C.A. Wood. The effects of shilling on final bid prices in online auctions. *Electronic Commerce Research and Applications*, 4:21–34, 2005.
9. V. Krishna. *Auction Theory.* Academic Press, 2002.
10. R. Preston McAfee. Mechanism design by competing sellers. *Econometrica*, 61(6):1281–1312, 1993.
11. M. Peters and S. Severinov. Competition among sellers who offer auctions instead of prices. *Journal of Economic Theory*, 75:141–179, 1997.
12. A. Rogers, E. David, and N.R. Jennings. Self organised routing for wireless micro-sensor networks. *IEEE Trans. on Systems, Man and Cybernetics: Part A*, 35(3):349–359, 2005.
13. W. Wang, Z. Hidvégi, and A.B. Whinston. Shill-proof fee (SPF) schedule: The sunscreen against seller self-collusion in online english auctions. Working Paper, 2004.

Robust Incentive-Compatible Feedback Payments

Radu Jurca and Boi Faltings

Ecole Polytechnique Fédérale de Lausanne (EPFL)
Artificial Intelligence Laboratory
CH-1015 Lausanne, Switzerland
{radu.jurca,boi.faltings}@epfl.ch

Abstract. Online reputation mechanisms need honest feedback to function effectively. Self interested agents report the truth only when explicit rewards offset the cost of reporting and the potential gains that can be obtained from lying. Payment schemes (monetary rewards for submitted feedback) can make truth-telling rational based on the correlation between the reports of different clients.

In this paper we use the idea of automated mechanism design to construct the best (i.e., budget minimizing) incentive-compatible payments that are also robust to some degree of private information.

1 Introduction

Online buyers increasingly resort to reputation forums for obtaining information about the products or services they intend to purchase. The testimonies of previous buyers disclose hidden, *experience-related* [13], product attributes (e.g., quality, reliability, ease of use, etc.) that can only be observed after the purchase. This previously unavailable information allows buyers to take better decisions.

Quality-based differentiation of products is also beneficial for the sellers. High quality, when recognizable by the buyers, brings higher revenues. Manufacturers can therefore optimally plan the investment in their products, such that the difference between the higher revenues of a better product, and the higher cost demanded by the improved quality, is maximized. Honest reputation feedback is thus essential for establishing an efficient market.

Human users exhibit high levels of honest behavior (and truthful sharing of information) without explicit incentives. However, in a future e-commerce environment dominated by rational agents acting to maximize their revenues, reputation mechanism designers need to make sure that sharing truthful information is in the best interest of the reporter.

Two factors make this task difficult. First, feedback reporting is usually costly. Users need to understand the rating scale, must fill in feedback forms and supervise the submission of the report. All these require the time and the conscious effort of the reporters. As feedback reporting does not bring direct benefits (the information is valuable only to subsequent buyers), rational agents are better off not to report at all.

Second, truth-telling is not always in the best interest of the reporter. In some settings, for instance, false denigration decreases the reputation of a product and allows the reporter to make a future purchase for a lower price. In other contexts, providers can offer monetary compensations in exchange for favorable feedback: e.g., doctors get gifts for recommending new drugs, authors ask their friends to write positive reviews about their latest book [6,16]. One way or another, external benefits can be obtained from lying and selfish agents will exploit them.

Both problems can be addressed by a payment scheme that explicitly rewards honest feedback by a sufficient amount Δ to offset both the cost of reporting and the gains that could be obtained through lying. Seminal work in the mechanism design literature [5,4] shows that side payments can be designed to create the incentive for agents to report their private opinions truthfully, a property called *incentive compatibility*. The best such payment schemes have been constructed based on "proper scoring rules" [11,7,2], and exploit the correlation between the observations of different buyers about the same good.

Miller, Resnick and Zeckhauser [12] adapt these results to online feedback forums. A central processing facility (the reputation mechanism) "scores" every submitted feedback by comparing it with another report (called the *reference* report) about the same good. They prove the existence of general incentive-compatible payment mechanisms where the return when reporting honestly is better by at least an arbitrary margin, Δ.

Jurca and Faltings [10] use an identical setting to apply *automated mechanism design* [3,15]. Incentive-compatible payments are computed by solving an optimization problem with the objective of minimizing the required budget. The simplicity of specifying payments through closed-form scoring rules is sacrificed for significant gains in efficiency.

Intuitively, payment mechanisms encourage truth-telling because reporters expect to get paid according to how well their feedback improves the current predictor of the reference report. Every feedback report modifies the reputation information, which acts as a predictor for future observations. The payment received by the reporter reflects the quality of the updated predictor, tested against the reference report. Assuming that the reference report is truthful, every agent naturally has the incentive to update the current reputation such that it mirrors her subjective beliefs. Thus agents report honestly, and truth-telling is a Nash equilibrium.

One key assumption behind such mechanisms is that the reputation mechanism and the reporters share the same prior information regarding the "reputation" of the rated product. Only in this case the honest report aligns the posterior reputation (as computed by the reputation mechanism) with the private posterior beliefs of the agent. When reporters have private prior information unknown to the reputation mechanism, it may be possible that some lying report maximizes the expected gain.

In this paper, we investigate feedback payment mechanisms that are incentive compatible even when reporters have some private information that is unknown to the reputation mechanism. Section 2 formally describes our setting. Section 3 describes the algorithm for computing the optimal incentive compatible payments with public prior information. Section 4 exemplifies what can go wrong if the reputation mechanism does not accurately know the beliefs of the reporters, followed by an algorithm, in Section 5, for computing incentive-compatible payments that are robust against a range of private beliefs. Finally we present future work and conclude.

2 The Setting

We consider an online market where a number of rational clients (or "agents") purchase the same product. The quality of the product remains fixed, and defines the product's (unknown) *type*. Let Θ be the finite set of possible types, and $\theta \in \Theta$ be a member of this set. Θ is common knowledge, and we assume that all clients share a common belief[1] regarding the prior probability $Pr[\theta]$, that the product is of type θ. $\sum_{\theta \in \Theta} Pr[\theta] = 1$.

Having purchased the product, clients perceive a noisy signal about the quality (i.e., true type) of the product. Let O^i denote the random signal observed by agent i, and let $\mathcal{S} = \{s_1, s_2, \dots s_M\}$ denote the set of possible values for O^i. The observations of different buyers are conditionally independent, given the type of the product. Let $f(s_j|\theta) = Pr[O^i = s_j|\theta]$ be the probability that a buyer observes the signal s_j when the true type of the product is θ. $f(\cdot|\cdot)$ is assumed common knowledge, and $\sum_{j=1}^{M} f(s_j|\theta) = 1$ for all $\theta \in \Theta$. We assume that different types generate different probability distributions for observable signals.

A central reputation mechanism asks each client to submit feedback. Let $a^i = (a^i_1, \dots, a^i_M)$ denote the reporting strategy of agent i, such that the agent will announce $a^i_j \in \mathcal{S}$ when her observed signal is s_j. The honest reporting strategy is denoted by $\bar{a} = (s_1, \dots, s_M)$, when the agent truthfully announces her observed signal.

The reputation mechanism pays clients for submitting feedback. The payment received by client i is computed by taking into account the signal announced by i, and the signal announced by another client, $r(i)$, called the reference reporter of i. Let $\tau(a^i_j, a^{r(i)}_k)$ be the payment received by agent i when she announces the signal a^i_j and $r(i)$ announces the signal $a^{r(i)}_k$. The expected payment of agent i depends on the prior belief, on her observation $O^i = s_j$, and on the reporting strategies a^i and $a^{r(i)}$:

$$V(a^i, a^{r(i)}|s_j) = E_{s_k \in \mathcal{S}}\big[\tau(a^i_j, a^{r(i)}_k)\big] = \sum_{k=1}^{M} Pr[O^{r(i)} = s_k|O^i = s_j]\tau(a^i_j, a^{r(i)}_k); \qquad (1)$$

The conditional probability distribution for the signal observed by the client $r(i)$ can be computed as:

$$Pr[s_k|s_j] = \sum_{\theta \in \Theta} f(s_k|\theta)Pr[\theta|s_j]; \qquad (2)$$

[1] This assumption is relaxed in Section 4.

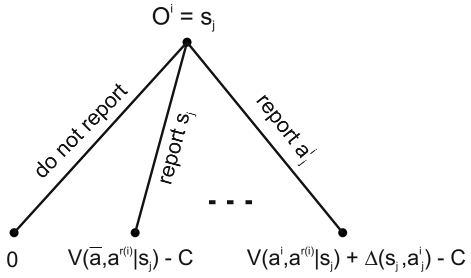


Fig. 1. Reporting feedback. Choices and Payoffs.

where $Pr[\theta|s_j]$ is the posterior probability of the type θ given the observation s_j, as given by Bayes' Law:

$$Pr[\theta|s_j] = \frac{f(s_j|\theta)Pr[\theta]}{Pr[s_j]}; \qquad Pr[s_j] = \sum_{\theta\in\Theta} f(s_j|\theta)Pr[\theta]; \tag{3}$$

All agents, as well as the reputation mechanism, compute these conditional probabilities in the same way.

Reporting feedback is expensive. Let C^i be the cost incurred by agent i for acquiring and reporting the observed signal. This cost is assumed independent of the beliefs and observations of the agent. Different agents can have different reporting costs; however, the variations are sufficiently small such that the upper bound, $C = \max_i C^i$, is finite and not "too far" from individual costs.

Agents can also obtain direct benefits from lying. Let $\Delta^i(s_j, a^i_j)$ be the external benefit agent i could obtain from reporting the signal a^i_j instead of s_j. As with the reporting cost, we assume there is an upper bound $\Delta(s_j, s_k) = \max_i \Delta^i(s_j, s_k)$ for the external benefit any agent could obtain by falsely reporting the signal s_k instead of s_j. For all signals $s_j \neq s_k \in \mathcal{S}$, $\Delta(s_j, s_j) = 0$ and $\Delta(s_j, s_k) \geq 0$.

3 Optimal Incentive Compatible Feedback Payments

Let us consider an agent i that purchases the product and observes the quality signal $O^i = s_j$. When asked by the reputation mechanism to submit feedback, the agent can choose: (a) to honestly report s_j, (b) to report another signal $a^i_j \neq s_j \in \mathcal{S}$ or (c) not to report at all. Figure 1 presents the agent's expected payoff for each of these cases, given the payment scheme $\tau(\cdot,\cdot)$ and the reporting strategy $a^{r(i)}$ of the reference reporter.

Truthful reporting is a Nash equilibrium (NEQ) if agent i finds it optimal to announce the true signal when the reference reporter also reports the truth. Formally, the honest reporting strategy $\bar{a}$ is a NEQ if and only if:

$$V(\bar{a}, \bar{a}|s_j) \geq V(a^*, \bar{a}|s_j) + \Delta(s_j, a^*_j);$$
$$V(\bar{a}, \bar{a}|s_j) \geq C;$$

for all signals $s_j \in \mathcal{S}$, and all reporting strategies $a^* \neq \bar{a}$. When the inequalities are strict, honest reporting is a strict NEQ.

For any observed signal $O^i = s_j \in \mathcal{S}$, there are $M-1$ different dishonest reporting strategies $a^* \neq \bar{a}$ that agent i can use: i.e., $a^*_j \in \mathcal{S} \setminus \{s_j\}$. Using (1) to expand the expected payment of a client, the NEQ conditions become:

$$\begin{aligned} \sum_{k=1}^{M} Pr[s_k|s_j]\Big(\tau(s_j,s_k) - \tau(s_h,s_k)\Big) > \Delta(s_j,s_h); \quad \forall s_j \neq s_h \in \mathcal{S} \\ \sum_{k=1}^{M} Pr[s_k|s_j]\tau(s_j,s_k) > C; \quad \forall s_j \neq s_h \in \mathcal{S} \end{aligned} \tag{4}$$

Any payment scheme $\tau(\cdot,\cdot)$ satisfying the conditions in (4) is incentive-compatible. [12] proves that such schemes exist.

Given the incentive-compatible payment scheme $\tau(\cdot,\cdot)$, the expected payment to an honest reporter is:

$$W = E_{s_j \in \mathcal{S}}\Big[V(\bar{a},\bar{a}|s_j)\Big] = \sum_{j=1}^{M} Pr[s_j]\Big(\sum_{k=1}^{M} Pr[s_k|s_j]\tau(s_j,s_k)\Big);$$

The optimal payment scheme minimizes the budget required by the reputation mechanism, and therefore solves the following linear program (i.e., linear optimization problem):

LP 1

$$\begin{aligned} min \quad & W = \sum_{j=1}^{M} Pr[s_j]\Big(\sum_{k=1}^{M} Pr[s_k|s_j]\tau(s_j,s_k)\Big) \\ s.t. \quad & \sum_{k=1}^{M} Pr[s_k|s_j]\Big(\tau(s_j,s_k) - \tau(s_h,s_k)\Big) > \Delta(s_j,s_h); \forall s_j \neq s_h \in \mathcal{S}; \\ & \sum_{k=1}^{M} Pr[s_k|s_j]\tau(s_j,s_k) > C; \quad \forall s_j \in \mathcal{S} \\ & \tau(s_j,s_k) \geq 0; \forall s_j, s_k \in \mathcal{S} \end{aligned}$$

The payment scheme $\tau(\cdot,\cdot)$ solving LP 1 depends on the cost of reporting, on the external benefits from lying, and on the prior belief about the type of the product. To illustrate these payments, let us consider a simple example.

Assume that the product purchased by the clients can be either *Good (G)* or *Bad (B)* (i.e., $\Theta = \{G,B\}$). The prior belief of the clients assigns the probabilities 0.8 and 0.2 to the product being good, respectively bad (i.e., $Pr[G] = 0.8, Pr[B] = 0.2.$). Clients perceive a binary quality signal (either *high* or *low* quality) once they purchase the product, such that $f(h|G) = 1 - f(l|G) = 0.9$ and $f(h|B) = 1 - f(l|B) = 0.2$. The probability that the next buyer experiences a high quality product is: $Pr[h] = 1 - Pr[l] = f(h|G)Pr[G] + f(h|B)Pr[B] = 0.76$.

The conditional probability distribution $Pr[O^{r(i)}|O^i]$ for the reference report follows from Bayes' law: $Pr[h|h] = 1 - Pr[l|h] = 0.86$ and $Pr[h|l] = 1 - Pr[l|l] = 0.43$. We take the reporting cost $C = 0.01$ (i.e., 1% of the normalized cost of

Table 1. Example. $Pr[G] = 0.8$, $Pr[h|h] = 0.86$ and $Pr[h|l] = 0.43$.

	$\tau(h,h)$	$\tau(h,l)$	$\tau(l,h)$	$\tau(l,l)$	
min	0.65	0.11	0.10	0.14	
s.t.	0.86	0.14	-0.86	-0.14	> 0.05
	0.86	0.14			> 0.01
	-0.43	-0.57	0.43	0.57	> 0.05
			0.43	0.57	> 0.01
	≥ 0	≥ 0	≥ 0	≥ 0	
Sol.	0.083	0	0	0.15	

the product) and the external benefits from lying $\Delta(l,h) = \Delta(h,l) = 0.05$. The optimization problem LP 1 is presented in Table 1, and defines the optimal payments: $\tau(h,h) = 0.083$, $\tau(l,l) = 0.15$, $\tau(h,l) = \tau(l,h) = 0$. The expected cost for the reputation mechanism (i.e., the expected payment to one agent) is 0.07 (i.e., 7% of the price of the product).

In [10] we show how to extend LP 1 in order to compute the optimal incentive compatible payments when:

– the available budget is fixed and the margins for truth-telling are maximized;
– the reputation mechanism can use several reference reports;
– the reputation mechanism may filter out some of the reports.

All resulting optimization problems are linear, and can be solved by polynomial[2] time algorithms. The resulting payments may decrease the budget required by the reputation mechanism up to one order of magnitude.

4 Honest Reporting with Unknown Beliefs

The optimal incentive-compatible payments computed in Section 3 rely on the posterior beliefs (i.e., the probabilities $Pr[s_k|s_j]$) of the reporters regarding the value of the reference reports. These can be computed by the reputation mechanism from:

– the prior belief, $Pr[\theta]$, that the product is of type θ,
– the conditional probabilities, $f(s_j|\theta)$, that a product of type θ generates the signal s_j,

using Bayes' Law as shown in the Eq. (2) and (3).

However, when the reporters have different beliefs regarding the values of the reference reports, the constraints in LP 1 do not accurately reflect the decision problem of the agents, and therefore, do not always guarantee an honest equilibrium.

Let us reconsider the example from Section 3, and the corresponding payments from Table 1. Assume an agent whose prior belief differs only slightly from that of

[2] In the size of the payment mechanism.

the reputation mechanism: e.g., $Pr[G]$, the probability that the product is *Good*, is 0.82 instead of 0.8, while other values remain the same. If the agent purchases a product of *low* quality, her private belief regarding the quality observed by the next buyer is $Pr^*[h|l] = 1 - Pr^*[l|l] = 0.45$ (instead of $Pr[h|l] = 0.43$ considered by the reputation mechanism). Simple arithmetics reveals that the agent is better off by reporting *high* instead of *low* quality: the expected gain from lying is $0.45 \cdot 0.083 + 0.55 \cdot 0 + \Delta(l, h) = 0.087$, while honest reporting brings only $0.45 \cdot 0 + 0.55 \cdot 0.15 = 0.082$.

4.1 Declaration of Private Information

To eliminate lying incentives, Miller et al. suggest that reporters should also declare their prior beliefs before submitting feedback [12]. The reputation mechanism could then use the extra information to compute the payments that makes truth-telling rational for every agent.

Unfortunately, such a mechanism can be exploited by self-interested agents when external benefits from lying are positive. Consider the same example as above: the agent has a prior belief which assigns probability 0.82 to the product being *Good*.

If the agent truthfully declares her prior belief, the reputation mechanism computes the optimal payments: $\tau(h, h) = 0.0841$, $\tau(l, l) = 0.1598$, $\tau(h, l) = \tau(l, h) = 0$, by solving LP 1. A truthful report following a negative experience (i.e., the agent observes and declares the signal l), is rewarded by an expected revenue equal to: $0.45 \cdot 0 + 0.55 \cdot 0.1598 = 0.0879$.

The agent can, however, declare the prior belief: $\overline{Pr}[G] = 1 - \overline{Pr}[B] = 0.11$. In this case, the payment scheme computed by the reputation mechanism will be: $\tau(h, h) = 0.2792$, $\tau(l, l) = 0.1375$, $\tau(h, l) = \tau(l, h) = 0$, and the optimal strategy for the client is to declare the signal h. The client's expected feedback payment thus becomes : $0.45 \cdot 0.2792 + 0.55 \cdot 0 + \Delta(l, h) = 0.1756 > 0.0879$, where $\Delta(l, h) = 0.05$ is the external revenue the agent can obtain by falsely declaring h instead of l.

The example provided above is, unfortunately, not unique. Profitable lying is possible because agents can find false prior beliefs that determine the reputation mechanism to compute feedback payments that make lying optimal. Thus, the agent obtains both the optimal feedback payment, and the external benefit from lying.

The false prior beliefs that make lying profitable can be easily computed based on the following intuition. The payment scheme defined by LP 1 makes it optimal for the agents to reveal their true *posterior* belief regarding the type of the product. When the prior belief is known, only the truly observed quality signal "aligns" the posterior belief of the reputation mechanism with that of the agent. However, when the prior belief must also be revealed, several combinations of prior belief and reported signal, can lead to the same posterior belief. Hence, the agent is free to chose the combination that brings the best external reward.

The false prior belief $(\overline{Pr}[\theta])_{\theta \in \Theta}$ and the false signal s_h that lead to the same posterior belief $(Pr[\theta|s_j])_{\theta \in \Theta}$, can be computed by solving the following system of linear equations:

$$\overline{Pr}[\theta|s_h] = \frac{f(s_h|\theta)\overline{Pr}[\theta]}{\sum_{t\in\Theta} f(s_h|t)\overline{Pr}[t]} = \frac{f(s_j|\theta)Pr[\theta]}{\sum_{t\in\Theta} f(s_j|t)Pr[t]} = Pr[\theta|s_j]; \quad \forall\theta \in \Theta; \qquad (5)$$

The system has $|\Theta|$ equations and $|\Theta|+1$ variables (i.e., the probabilities $\overline{Pr}[\theta]$ and the signal s_h); therefore, there will generally be several solutions that make lying profitable. The agent may choose the one that maximizes her expected payment by solving the following nested linear problem:

$$\begin{aligned} max \quad & \Delta(s_j, s_h) + \sum_{k=1}^{M} Pr[s_k|s_j]\tau(s_h, s_k) \\ s.t. \quad & \overline{Pr}[\theta|s_h] = Pr[\theta|s_j];\ \forall\theta \in \Theta; \\ & \tau(\cdot,\cdot) \text{ solves LP 1 for the prior beliefs } \overline{Pr}[\theta] \end{aligned}$$

To enforce truth-telling, Prelec [14] suggests payments that also depend on the declared priors. Agents are required to declare both the observed signal, and a prediction of the signals observed by the other agents (which indirectly reflects the agent's private information). The proposed *"truth serum"* consists of two additive payments: an *information* payment that rewards the submitted report, and a *prediction* payment that rewards the declared private information. Prelec shows that honesty is the highest paying Nash equilibrium. Nonetheless, his results rely on the assumption that a prior probability distribution over all possible private beliefs (not the belief itself) is common knowledge.

Another solution has been suggested by Miler et al. in [12]. Miss-reporting incentives can be eliminated if agents declare their prior beliefs before the actual interaction takes place. As posteriors are not available yet, the agent cannot manipulate the declared prior belief in order to avoid the penalty from lying. However, such a process has several practical limitations.

First, the enforcement of prior belief declaration before the interaction can only be done if a central authority acts as an intermediary between the buyer and the seller. The central proxy may become a bottleneck and adds to the transaction cost. Second, the declaration of prior beliefs could significantly delay the access to the desired good. Finally, the reporting of priors adds to the reporting cost (reporting probability distributions is much more costly than reporting observed signals) and greatly increases the budget required by an incentive-compatible mechanism.

5 Robust Incentive Compatible Payments

In this paper we pursue an alternative solution for dealing with unknown beliefs. We start from the assumption that the private beliefs of most rational agents will not differ significantly from those of the reputation mechanism. The beliefs of the reputation mechanism, as reflected in the publicly available reputation information, have been constructed by aggregating all feedback reports submitted by all previous users. Assuming that agents trust the reputation mechanism to publish truthful information, their private information will trigger only marginal changes to the beliefs. Thus, rather than build a system that can accommodate all private beliefs, we focus on mechanisms that are incentive-compatible for most priors, i.e., the priors within certain bounds from those of the reputation mechanism.

Let $(Pr[\theta])_{\theta\in\Theta}$ characterize the prior belief of the reputation mechanism and let $(Pr^*[\theta] = Pr[\theta]+\varepsilon_\theta)_{\theta\in\Theta}$ be the range of private beliefs the clients might have, where: $\sum_{\theta\in\Theta}\varepsilon_\theta = 0$, and $\max(-\epsilon, -Pr[\theta]) \le \varepsilon_\theta \le \min(\epsilon, 1-Pr[\theta])$, $\epsilon > 0$.

Replacing the private beliefs in (2) and (3), the conditional probabilities for the reference rater's signals become:

$$Pr^*[s_k|s_j] = \frac{\sum_{\theta\in\Theta} f(s_k|\theta)f(s_j|\theta)(Pr[\theta]+\varepsilon_\theta)}{\sum_{\theta\in\Theta} f(s_j|\theta)(Pr[\theta]+\varepsilon_\theta)}; \tag{6}$$

Let $Pr^*_m[s_k|s_j]$ and $Pr^*_M[s_k|s_j]$ be the minimum, respectively the maximum values of $Pr^*[s_k|s_j]$ as the variables $(\varepsilon_\theta)_{\theta\in\Theta}$ take values within the acceptable bounds. If we modify LP 1 such that the constraints on the optimal payments are satisfied for all acceptable values of $Pr^*[s_k|s_j]$, we obtain a payment scheme that is incentive compatible for all private beliefs that are not *too far* from the belief of the reputation mechanism.

Representing linear constraints for a continuous range of parameters is not accepted by linear solvers. The constraint:

$$\sum_{k=1}^{M} Pr^*[s_k|s_j]\Big(\tau(s_j,s_k) - \tau(s_h,s_k)\Big) > \Delta(s_j,s_h); \tag{7}$$

is satisfied for all possible values of $Pr^*[s_k|s_j] \in \Big[Pr^*_m[s_k|s_j], Pr^*_M[s_k|s_j]\Big]$, only when:

$$\min_{Pr^*[s_k|s_j]}\left(\sum_{k=1}^{M} Pr^*[s_k|s_j]\Big(\tau(s_j,s_k) - \tau(s_h,s_k)\Big)\right) > \Delta(s_j,s_h); \tag{8}$$

If the probabilities $Pr^*[s_k|s_j]$ were independent,[3] the minimum would be given by one of the combinations of extreme values: i.e., $Pr^*[s_k|s_j]$ equal either $Pr^*_m[s_k|s_j]$ or $Pr^*_M[s_k|s_j]$. Therefore, by replacing every constraint (7), with 2^M linear constraints, one for every combination of extreme values of $Pr^*[s_k|s_j]$, we impose stricter condition than (8). The optimization problem defining the payment scheme is similar to LP 1, where every constraint has been replaced by 2^M linear constraints, one for every combination of extreme values of $Pr^*[s_k|s_j]$.

We evaluated experimentally the effect of private beliefs on the expected cost of the incentive-compatible mechanism. For that purpose, we generated 2000 random problems as described in Appendix A. For each problem, we took different tolerance levels to private beliefs (i.e., $\epsilon = \{0, 0.02, 0.05, 0.07, 0.1\}$) and solved the linear optimization problem that defines the robust, incentive compatible payments. We used average hardware (e.g., Pentium Centrino 1.6MHz,

[3] They are not, because they are connected through the same variables (ε_θ).

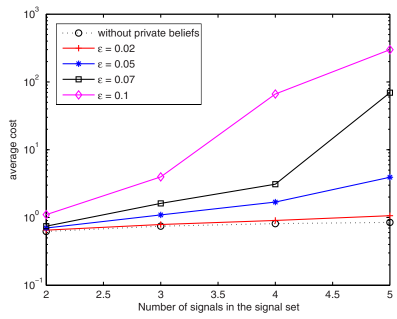


Fig. 2. Average expected payment to one agent for different tolerance levels of private beliefs

Table 2. Average CPU time (and standard deviation) for computing the optimal payment scheme with private beliefs

M	CPU time [ms]	Std. dev. [ms]
2	14.117	4.9307
3	38.386	4.1765
4	485.33	50.546
5	798.28	722.5

1Gb RAM, WinXP) and the CPLEX[4] linear solver. Table 2 presents the average CPU time required for computing the payments. Due to the exponential number of constraints, the time required to compute the optimal payments increases exponentially with the number of signals, M. For $M = 6$ signals, the computation already takes more than one second.

Figure 2 presents the average cost of an incentive-compatible payment scheme that tolerates private beliefs. We plot the average expected payment to one agent for different number of signals, and different tolerance levels for private beliefs. The cost of the mechanism increases quickly with ϵ, the tolerated range of beliefs. For beliefs within $\pm 10\%$ of those of the reputation mechanism, the cost of the mechanism increases one order of magnitude. Note, however, that the constraints defining the payment scheme are stricter than necessary. As future research, we intend to define non-linear algorithms that can approach the truly optimal payments.

5.1 General Tolerance Intervals for Private Information

Instead of modeling private information as small perturbations to the prior belief regarding the true type of the product, we consider in this section a more general case, where the conditional probabilities $Pr[s_k|s_j]$ that parameterize LP 1 are allowed to vary within certain limits. Such variations can account for various sources of private information: e.g., private beliefs regarding the true type of the product, private information regarding the conditional distribution of signals, or even small changes to the true type of the product. This approach is similar to the work of Zohar and Rosenschein [17].

Without modeling the real source of private information, we assume that the conditional probability distributions $Pr^*[\cdot|s_j]$ (for all s_j) are not too far from the probability distributions $Pr[\cdot|s_j]$ computed by the reputation mechanism. We will use the L_2 norm for computing the distance, and assume that:

[4] www.ilog.com

$$\sum_{k=1}^{M}\Big(Pr^*[s_k|s_j]-Pr[s_k|s_j]\Big)^2 \leq \varepsilon^2;\ \forall s_j\in\mathcal{S}; \tag{9}$$

for some positive bound ε. The incentive-compatibility constraints must enforce that for any value of the probabilities $Pr^*[\cdot|\cdot]$, honesty gives the highest payoff. Formally,

$$\min_{Pr^*[\cdot|\cdot]}\left(\sum_{k=1}^{M}Pr^*[s_k|s_j]\Big(\tau(s_j,s_k)-\tau(s_h,s_k)\Big)\right) > \Delta(s_j,s_h);\ \forall s_j\neq s_h\in\mathcal{S};$$
$$s.t. \sum_{k=1}^{M}\Big(Pr^*[s_k|s_j]-Pr[s_k|s_j]\Big)^2\leq\varepsilon^2;$$

This optimization problem can be solved analytically by writing the Lagrangian and enforcing the first order optimality conditions. We thus obtain:

$$\min\left(\sum_{k=1}^{M}Pr^*[s_k|s_j]\Big(\tau(s_j,s_k)-\tau(s_h,s_k)\Big)\right)=$$
$$\sum_{k=1}^{M}Pr[s_k|s_j]\Big(\tau(s_j,s_k)-\tau(s_h,s_k)\Big)-\varepsilon\sqrt{\sum_{k=1}^{M}\Big(\tau(s_j,s_k)-\tau(s_h,s_k)\Big)^2};$$

and the best (i.e., cheapest) incentive compatible payments that are robust to private information (i.e., have robustness level ε^2) are obtained by solving the conic optimization problem:

CP 1

$$\begin{aligned}
&min\quad W=\sum_{j=1}^{M}Pr[s_j]\Big(\sum_{k=1}^{M}Pr[s_k|s_j]\tau(s_j,s_k)\Big)\\
&s.t.\quad \sum_{k=1}^{M}Pr[s_k|s_j]\Big(\tau(s_j,s_k)-\tau(s_h,s_k)\Big)-\varepsilon\sqrt{\sum_{k=1}^{M}\Big(\tau(s_j,s_k)-\tau(s_h,s_k)\Big)^2}>\Delta(s_j,s_h);\\
&\qquad\ \sum_{k=1}^{M}Pr[s_k|s_j]\tau(s_j,s_k)-\varepsilon\sqrt{\sum_{k=1}^{M}\tau(s_j,s_k)^2}>C;\\
&\qquad\ \forall s_j\neq s_h\in\mathcal{S};\quad \tau(s_j,s_k)\geq 0;\forall s_j,s_k\in\mathcal{S}
\end{aligned}$$

where $Pr[\cdot|\cdot]$ are the probabilities computed by the reputation mechanism. Such problems can be solved by polynomial time algorithms.

We evaluated experimentally the cost of general private information as reflected on the expected payment to one reporter. As in the previous section, we generated 2000 random problems (details in Appendix A) and for different levels of robustness, we solved CP 1 to obtain the robust incentive compatible payments. Table 3 presents the average CPU time required to compute the payments. As expected, the values are much smaller than those of Table 2.

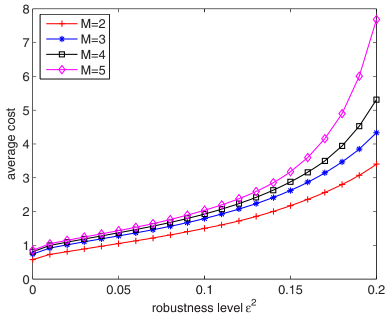


Fig. 3. Average expected payment to one agent for different levels of robustness to general private information

Table 3. Average CPU time (and standard deviation) for computing the optimal payment scheme with general private information

M	CPU time [ms]	Std. dev. [ms]
2	3.46	24.40
3	7.31	9.04
4	16.05	8.07
5	44.89	15.74

Figure 3 plots the average expected payment to one agent for different number of signals, and different tolerance levels to private information. Like in Figure 2, the cost of the mechanism increases exponentially with the robustness level, ε^2.

One important remark about the results of this section is that agents trust the reputation mechanism to publish truthful information. Only in this case agents are likely to adopt (with marginal changes) the beliefs of the reputation mechanism, and have incentives to report honestly. While the trustworthy of the reputation mechanism is an interesting topic on its own, let us note that agents can verify[5] whether or not the payments advertised by the reputation mechanism actually "match" the beliefs and the robustness bound published by the reputation mechanism. On the other hand, the payments that match the true beliefs of the reputation mechanism are the smallest possible, as guaranteed by the corresponding optimization problems.

However, understanding exactly what the reputation mechanism can do in order to manipulate reputation information without being detected, while still providing decent honest reporting incentives requires further work.

6 Discussion

We have assumed in our paper that the true quality of the product (i.e., the true type) is fixed. In real settings, however, quality does change either because the technology evolves, or because initial defects get identified and corrected. Under such circumstances, it becomes even more important to design payments that are robust to a wide range of private beliefs: incremental changes in the true quality will fall within the tolerance levels of the payment scheme and do not shift reporting incentives. However, smarter payments may be capable of modeling quality change, and factor it appropriately in reporting incentives. This remains as future work.

[5] By checking that the payments $\tau(\cdot,\cdot)$ solve the optimization problems LP 1 or CP 1.

The reporting and honesty costs enclose a powerful framework for treating the collusion between buyers an product manufacturers. Dishonest feedback from clients usually creates advantages for product manufacturers (false positive feedback is beneficial for the product's owner, false negative feedback benefits the competitors). The collusion between clients and providers becomes interesting when the benefit obtained by providers from a false report offsets the payment returned to the client in exchange for lying. The feedback payments presented in this paper make sure that no provider can afford to buy false reports; the collusion thus becomes irrational.

The honest reporting Nash Equilibrium is unfortunately not unique. Other lying equilibria exist, and some of them generate higher expected payoffs for reporters than the truthful one. In a previous result, [8] we show that a small number of *trusted reports* (i.e., feedback reports that are true with high probability) can eliminate (or render unattractive) lying Nash equilibria. As future work, we intend to extend the presented framework to also account for multiple equilibria.

Collusion between clients remains a problem for this class of incentive compatible mechanisms. Agents can synchronize their possibly false reports in order to increase their revenue. Choosing randomly the reference report for every submitted feedback can help eliminate small coalitions: only large coalitions are rational, such that the probability of having a reference report from the same coalition is big enough. Another safeguard against reporting coalitions is to use trusted reports. In some settings [9], a small number of trusted reports can make collusion irrational.

One interesting direction for future research is to design mechanisms that can better tolerate private beliefs. As discussed in Section 4, our algorithms generate payments that increase exponentially with the range of tolerated private information. However, using a combination of different techniques (e.g., the payments also depend on declared priors, priors may be discounted as they diverge from the belief of the reputation mechanism) may result in cheaper mechanism.

7 Conclusion

Honest feedback is essential for the effectiveness of online reputation mechanisms. When feedback reporters are self-interested, explicit payments can make truthful reporting rational. Most of the existing incentive-compatible payment schemes are constructed based on proper scoring rules. Lately, computational techniques based on the idea of *automated mechanism design* have made it possible to significantly decrease the cost of incentive-compatibility by computing the best payment scheme for each context.

In the current paper we extent this line of research, by studying incentive-compatible payments that are also robust to some degree of private information. We show how the smallest amount of private information (possessed by the agents, and unknown to the reputation mechanism) can disrupt the truth-telling incentives provided by traditional payment mechanisms. As a consequence, we suggest the automated design of robust payments that are incentive-compatible for a range of beliefs. The resulting optimization problems are more complex, but can still be solved efficiently for practical settings.

References

1. A. Ben-Tal and A. Nemirovski. Robust Solutions of Uncertain Linear Programs. *Op. Research Letters*, 25:1–13, 1999.
2. R. T. Clemen. Incentive contracts and strictly proper scoring rules. *Test*, 11: 167–189, 2002.
3. V. Conitzer and T. Sandholm. Complexity of mechanism design. In *Proceedings of the Uncertainty in Artificial Intelligence Conference (UAI)*, 2002.
4. J. Crémer and R. P. McLean. Optimal Selling Strategies under Uncertainty for a Discriminating Monopolist When Demands Are Interdependent. *Econometrica*, 53(2):345–61, 1985.
5. C. d'Aspremont and L.-A. Grard-Varet. Incentives and Incomplete Information. *Journal of Public Economics*, 11:25–45, 1979.
6. A. Harmon. Amazon Glitch Unmasks War of Reviewers. *The New York Times*, February 14, 2004.
7. S. Johnson, J. Pratt, and R. Zeckhauser. Efficiency Despite Mutually Payoff-Relevant Private Information: The Finite Case. *Econometrica*, 58:873–900, 1990.
8. R. Jurca and B. Faltings. Enforcing Truthful Strategies in Incentive Compatible Reputation Mechanisms. In *Internet and Network Economics*, volume 3828 of *LNCS*, pages 268 – 277. 2005.
9. R. Jurca and B. Faltings. Reputation-based Service Level Agreements for Web Services. In *Service Oriented Computing (ICSOC - 2005)*, volume 3826 of *LNCS*, pages 396 – 409. 2005.
10. R. Jurca and B. Faltings. Minimum payments that reward honest reputation feedback. In *Proceedings of the ACM Conference on Electronic Commerce*, Ann Arbor, Michigan, USA, June 11-15 2006.
11. M. Kandori and H. Matsushima. Private observation, communication and collusion. *Econometrica*, 66(3):627–652, 1998.
12. N. Miller, P. Resnick, and R. Zeckhauser. Eliciting Informative Feedback: The Peer-Prediction Method. *Management Science*, 51:1359 –1373, 2005.
13. A. Parasuraman, V. Zeithaml, and L. Berry. A Conceptual Model of Service Quality and Its Implications for Future Research. *Journal of Marketing*, 49:41–50, 1985.
14. D. Prelec. A bayesian truth serum for subjective data. *Science*, 306(5695):462–466, 2004.
15. T. Sandholm. Automated mechanism design: A New Application Area for Search Algorithms. In *Proceedings of the International Conference on Principles and Practice of Constraint Programming*, 2003.
16. E. White. Chatting a Singer Up the Pop Charts. *The Wall Street Journal*, October 15, 1999.
17. A. Zohar and J. Rosenschein. Robust Mechanisms for Information Elicitation. In *Proceedings of the AAAI*, 2006.

A Generating Random Settings

We consider settings where M possible product types are characterized each by one quality signal: i.e., the sets $\mathcal{S}$ and Θ have the same number of elements, and every type $\theta_j \in \Theta$ is characterized by one quality signal $s_j \in \mathcal{S}$. The conditional probability distribution for the signals observed by the buyers is computed as:

$$f(s_k|\theta_j) = \begin{cases} 1-\epsilon & \text{if} \quad k=j; \\ \epsilon/(M-1) & \text{if} \quad k \neq j; \end{cases}$$

where ϵ is the probability that a client misinterprets the true quality of the product (all mistakes are equally likely). We take $\epsilon = 10\%$.

The prior belief is randomly generated in the following way: for every $\theta_j \in \Theta$, $p(\theta_j)$ is a random number, uniformly distributed between 0 and 1. The probability distribution over types is then computed by normalizing these random numbers: $Pr[\theta_j] = \frac{p(\theta_j)}{\sum_{\theta \in \Theta} p(\theta)}$; The external benefits from lying are randomly uniformly distributed between 0 and 1.

The CrocodileAgent 2005: An Overview of the TAC SCM Agent

Ana Petric, Vedran Podobnik, and Gordan Jezic

University of Zagreb
Faculty of Electrical Engineering and Computing / Department of Telecommunications
HR-10000 Zagreb, Croatia
{ana.petric,vedran.podobnik,gordan.jezic}@fer.hr
http://www.tel.fer.hr

Abstract. The Trading Agent Competition (TAC) is an international forum which promotes high quality research regarding the trading agent problem. One of the TAC competitive scenarios is Supply Chain Management (SCM) where six agents compete by buying components, assembling PCs from these components and selling the assembled PCs to customers. In this paper, we describe the strategies implemented in the CrocodileAgent, our entry in 2005 TAC SCM. We describe the structure and functionalities of the CrocodileAgent, the implementation of the basic agent tasks, and algorithms for ordering components and determining the profit margin. The agent's performances in the 2005 TAC SCM competition, as well as in a series of controlled experiments, are discussed.

1 Introduction

The advent of the Internet enabled proliferation of the electronic commerce (e-commerce), what has made an immense effect on the manner in which both the businesses and the consumers buy and sell goods. While the initial architecture of the Web was geared towards delivering information visually to humans, currently we are witnessing a transformation of the Internet into environment filled with goal directed applications that intelligibly and adaptively coordinate information exchanges and actions (Web 2.0 and Web 3.0) [1]. Consequently, the Internet becomes an enabler of the digital economy, which provides a new level and form of connectivity among multiple heterogeneous entities, giving rise to a vast new range of business combinations [2]. Additionally, by utilizing the technology of intelligent software agents [1, 3, 4, 5], the digital economy automates business transactions.

Supply chain management (SCM) coordinates planning and organizing activities across the multiple entities involved in the manufacturing and delivering a product or creating and provisioning a service. These activities generally include procurement of needed resources, their transformation into intermediate subassemblies or final products/services and delivery/provisioning to consumers. As today's global economy gears towards outsourcing, efficient SCM mechanisms become essential for enhancing businesses' agility [6]. Global competition in real-world markets creates great pressure on businesses to continuously reduce operating expenses and increase profitability, while enhancing consumer responsiveness and shortening lead-times.

In the paper we describe the CrocodileAgent 2005, an intelligent agent we developed to participate in 2005 TAC SCM (Trading Agent Competition Supply Chain Management) competition. The paper is organized as follows. Section 2 briefly presents the TAC SCM game. An overview of the CrocodileAgent's architecture and functionalities is given in Section 3. Section 4 comments the agent's ranking in the 2005 TAC SCM competition. Section 5 elaborates the results of an experiment we conducted in our laboratory and includes a detailed analysis of the results. In Section 6 the directions for future work are briefly proposed.

2 The TAC SCM Game

The Trading Agent Competition is an international forum that promotes high-quality research on the trading agent problem. One of its game scenarios is Supply Chain Management, which is presented in the Figure 1. In the TAC SCM game [7, 8], each of the six agents included in the game has its own personal computer (PC) manufacturing company. During the 220 TAC days, agents compete in two different markets. In *B2B market*, TAC SCM Agents compete in buying resources necessary to produce PCs. Participants in this market are six agents and eight suppliers which produce four types of components (CPUs, motherboards, memory, hard drives) with different performances. In its factory, an agent can manufacture 16 different types of PCs divided into three market segments. In the *B2C market*, TAC SCM Agents try to sell all the PCs they produced to customers and, at the same time, earn as much money as possible. Participants in this market are six agents and hundreds of customers with varying demand and reserve prices they are willing to pay for the PCs.

The aim of the TAC SCM game is to explore how to maximize the profit in the stochastic environment of volatile market conditions. Therefore, it is important to develop an agent capable of reacting quickly to changes taking place during the game. Furthermore, it is critical to implement predictive mechanisms which enable an agent's proactive behaviour. The idea is to build a robust, highly-adaptable and easily-configurable mechanism for efficiently dealing with all SCM facets, from resource procurement and inventory management to goods production and delivery/provisioning [9]. The key element in creating a profitable SCM strategy is to achieve successful balancing of reliable consumer delivery with manufacturing and inventory management costs [7]. The greatest challenges which are put in front of SCM strategy designers are *problem complexity*, *environment stochasticity* and *competitive ambient*. Problem complexity derives from the vast number of actors that continuously request or provide various artefacts, thus creating numerous possible business combinations in the time-restricted environment. Stochasticity is consequence of fact that TAC SCM Agents have access only to incomplete and imperfect information. Competitive ambient is a result of the fact that all the e-commerce actors want to maximize their utility functions.

The TAC SCM system constitutes an environment suitable for testing various strategies for ensuring efficient, flexible and dynamic SCM, which is vital for the competitiveness of manufacturing enterprises as it directly impacts their ability to meet changing market demands in a timely and cost effective manner [10]. TAC SCM

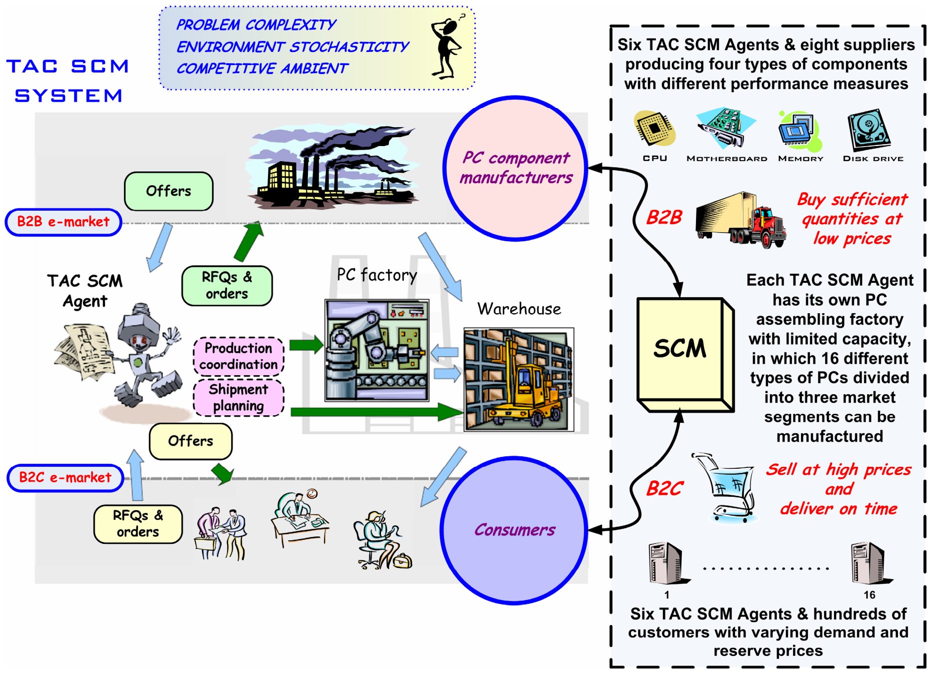

TAC SCM SYSTEM
PROBLEM COMPLEXITY
ENVIRONMENT STOCHASTICITY
COMPETITIVE AMBIENT
B2B e-market
Offers
RFQs & orders
TAC SCM Agent
Production coordination
Shipment planning
PC factory
Warehouse
PC component manufacturers
Consumers
Offers
B2C e-market
RFQs & orders
Six TAC SCM Agents & eight suppliers producing four types of components with different performance measures
CPU
MOTHERBOARD
MEMORY
DISK DRIVE
B2B
Buy sufficient quantities at low prices
SCM
Each TAC SCM Agent has its own PC assembling factory with limited capacity, in which 16 different types of PCs divided into three market segments can be manufactured
B2C
Sell at high prices and deliver on time
1
16
Six TAC SCM Agents & hundreds of customers with varying demand and reserve prices

Fig. 1. The relationships in the TAC SCM system

tournaments also provide an opportunity to analyze effects common in real-world business transacting, such as the bullwhip effect, and its relationship with company profits [11]. Furthermore, the tournament can help in developing methods for identifying the current economic regime and forecasting market changes [12].

3 The CrocodileAgent

The CrocodileAgent [4, 13] is an intelligent agent developed at the Department of Telecommunications, Faculty of Electrical Engineering and Computing, University of Zagreb, Croatia. Designing the CrocodileAgent was an extension of a project that started in 2004 when we developed the KrokodilAgent [14] and for the first time joined the TAC SCM competition.The CrocodileAgent was developed directly from the KrokodilAgent by performing iterative changes. Results of these changes and the interdependence of all the implemented algorithms were carefully monitored after each iteration. At the end of the development process, a new agent was created.

The CrocodileAgent inherited the basic architecture from its forerunner. This is understandable since it was developed by evolving the KrokodilAgent. Figure 2 shows the basic agent architecture. In order to enable easier development and transparent implementation of agent functionalities, the CrocodileAgent was divided into five logical parts: CustomerImpl, FactoryImpl, InventoryImpl, SupplierImpl and ZTELAgent. The central component of the agent is the ZTELAgent which inherits the

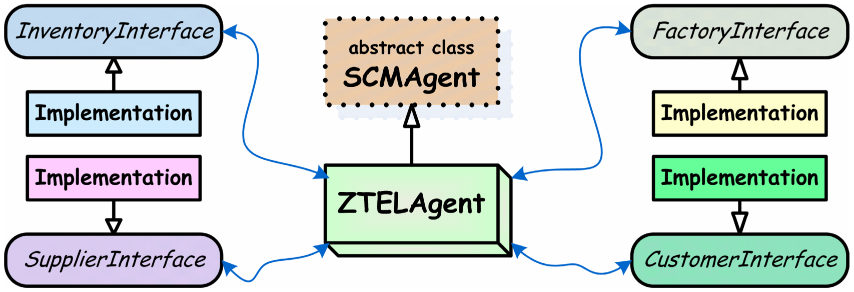


Fig. 2. The CrocodileAgent's architecture

SCMAgent class. The SCMAgent is an abstract class which defines all the methods that an agent needs to support in order to participate in the TAC SCM game. The ZTELAgent coordinates collaboration between all the other Agent components; it is responsible for the simulation status, the start and the end of the game.

The CustomerImpl is responsible for the agent's interaction with customers. The InventoryImpl's task is to provide enough components so that PC production can go on without any disturbances. Furthermore, the InventoryImpl keeps track of component prices. The SupplierImpl sends orders to the suppliers and takes care of delivered components. The agent's daily responsibilities are: negotiating supply contracts, bidding for customers' orders, managing daily assembly activities and shipping completed orders to customers. These tasks will to be described in the following sections.

3.1 Negotiation of Supply Contracts

There are two basic supply tactics: day-0 component procurement and ordering components during the game. In other words, the agent ordered a large number of components on day-0, and then if the need for components occurred during the game, additional components were ordered. This combination was very successful in the 2004 TAC SCM competition. The agent bought cheap components because there was no prior component demand.

The first step in the development of the CrocodileAgent was to modify the supply procurement mechanism. The reason for this was a change in the game rules regarding the suppliers and the way they calculate component prices. The CrocodileAgent was modified to send two RFQs (*Requests for Quotes*) requesting the complete amount of the needed component and accept only the cheaper one. In such a manner a very flexible mechanism for supply procurement was created and, with smaller adjustments, was held until the end of the competition. A description of the CrocodileAgent's supply tactics follows.

3.1.1 Day-0 Procurement

The agent sends RFQs with the following delivery dates: 3, 9, 17, 27 and 69. The reserve prices the agent is willing to pay for the components are 102%, 107%, 92%, 82% and 77% of the nominal price on the respective delivery dates. The referred parameters were determined after conducting a series of experiments. The requested quantities are smaller in short-term RFQs and larger in long-term RFQs. In case that a certain component is produced by two suppliers, RFQs are always sent to both suppliers that produce the needed component. The agent's strategy of accepting the cheaper offer causes no permanent damage, just a temporary droop of the agent's reputation in the eyes of the supplier whose offer was not accepted. Since the requested quantities are not high, the agent's reputation fully recovers in 20-30 days.

Unfavourable situation happens when the chosen supplier can not deliver the requested quantity on time. In that case the agent accepts partial offers. This way, the agent gets a smaller amount of components than planed. To avoid potential component shortage caused by the supplier's cancellation, the agent modifies the quantities and reserve prices for more aggressive purchasing in the near future.

3.1.2 Component Purchase During the Game

Some of the parameters used in component purchase are:

N_{min} – the minimal quantity of components required to be in storage,
N_{max} – the maximal quantity of components allowed in storage,
N_{ord} – the maximal amount of components that can be ordered each day,
N_{tdy} – the quantity of a certain component used in PC production per day,
N_{inv} – the number of components currently stored in the warehouse.

As the end of the game approaches, values of the parameters N_{min}, N_{max} and N_{ord} are lowered accordingly. At the beginning of each day, the agent calculates the component quantity ordered, but not delivered, up to that moment for each component separately. Since the components with an earlier delivery date have higher priority, the agent's ordered quantities of components are multiplied with a distance factor. The distance factor is a value between 0 and 1; the factor shrinks from 1 to 0 as the delivery date grows. When the delivery date reaches 30 days (from the current day) the distance factor becomes 0. The parameter obtained by performing this calculation is referred to as the *evaluatedQuantity*. Similarly, we calculate the *evaluatedLongTermQuantity* which represents the quantity of all the ordered components that have a delivery date higher than 33 days.

For each component, the agent checks to see if the following condition fulfilled:

$$N_{inv} + \mathit{evaluatedQuantity} \geq N_{max}\,. \tag{1}$$

The components are not ordered if condition (1) is fulfilled. The agent counts the number of days in a row that the component is not ordered. In the first part of the game, if the component purchase is inactive for 5 days in a row and if the *evaluatedLongTermQuantity* is lower than its upper limit in spite of condition (1), the agent sends long-term RFQs to ensure cheap components for the second part of the game. If condition (1) is not fulfilled, the following condition is considered:

$$N_{inv} + N_{tdy} > N_{min}\,. \tag{2}$$

If condition (2) is not fulfilled the agent purchases components more aggressively with the purpose of getting the number of components in the warehouse above N_{min} as soon as possible. Otherwise, he sends three RFQs with the purpose of maintaining the present quantity of components in the warehouse. Regardless of condition (2), the agent sends two long-term RFQs to ensure long-term occupancy of the warehouse. It is important to point out that these are only the main characteristics of the algorithm. Additionally, there are special mechanisms which calculate the reserve prices and exact quantities that need to be ordered. A simplified description of some of these mechanisms follows:

- The *lowComponentAlarm* contains several levels and marks the very low quantity of a certain component in the warehouse. In case the alarm is set, the agent is allowed to pay a higher price than usual for the component. Different components have different prices so the maximal reserve price for processors it is 115% of their nominal price while for other components it is 130% of their nominal price.
- The *demandPurchaseQuantityFactor* is modified only according to customer demand. Sometimes during the game, the customer demand rises rapidly. When this happens the agent uses more components to produce more PCs, so the parameter is increased to ensure that the agent does not run out of components and even more important, looses potentially profitable PC orders.
- The *componentsInInventoryShortageFactor* measures the successfulness of the day-0 purchase. If the purchase was not successful for a certain component, the agent is allowed to order larger quantities in the first phase of the game. The influence of this factor fully disappears after day 50.

Special attention was paid to the end of the game. The intention was to maintain a low level of all components in the warehouse until the game ended to enable the agent to send offers to customers for as long as possible. It is important to allow the agent to make short-term procurement of any type of component to prevent the situation were a large quantity of one component is left over because the agent had spent all the other components needed to produce a certain type of PC.

3.2 Bidding for Customer Orders

In the first version, the CrocodileAgent sent an offer only if he had enough components to produce all the PCs that were requested in the RFQ. This tactic is good because it prevents the situation where the agent has to pay penalties for delivering PCs late or not delivering them at all. The agent can also send offers for RFQs with short delivery dates because the factory can produce the PCs the following day. This tactic, however, has its disadvantages. Since the agent does not get orders for all the offers sent, a part of the agent's factory capacities remain unutilized and some of the components stay unused for some time. This problem was solved by sending more offers than before. The number of sent offers was carefully calculated based on the number of components in the warehouse and current customer demand.

After analyzing the TAC SCM 2005 competition we established that the agent's selling side was actually its weaker side. One of the most important reasons is the fact that the CrocodileAgent does not use any prediction methods when selling PCs. Most of its decisions are based on the past and present state of the PC market.

3.2.1 An Algorithm for Sending Offers

We now describe the main features of the agent's algorithm for sending offers to customers. The agent gives each RFQ a grade from 1 to 10, where a higher grade marks more profitable RFQs; within those grades the RFQs are sorted in chronological order of their delivery dates. The grade is determined by the difference between the customer's reserve price and the agent's cost of producing that PC. After grading and sorting the RFQs the agent starts to send offers if two conditions are fulfilled:

– There are enough components to produce the requested PCs. Note, however, that the number of offered PCs is actually a little bit higher than the number of PCs that the agent can produce from components located in the warehouse - recall that the number of PCs that are offered is carefully calculated,
– The agent's PC production cost is lower than the customer's reserve PC price.

In case one of these two conditions is not fulfilled, the agent checks if the requested PCs can be delivered form the reserve PCs stored in the warehouse.

This algorithm comes in three versions: *handleCustomerRFQsNormal*, *handleCustomerRFQsHighDemand* and *handleCustomerRFQsEndGame*. The version that is active on a certain day is determined depending on the number of production cycles needed to produce all the active orders and the algorithm that was used the day before and the stage of the game (beginning, middle, end). The most important difference between these three algorithms is the agent's method for calculating the PC offer prices. Since *handleCustomerRFQsHighDemand* and *handleCustomerRFQsEndGame* algorithms are designed for special cases, most of the time during game the agent uses *handleCustomerRFQsNormal* algorithm. It determines the offer price in two ways:

– The offer price is slightly under the customer's reserve price in case the agent offers PCs that are already produced and stored in the warehouse.
– In case the agent has to manufacture the PCs first, the offer price is calculated as the basic PC price (sum of the average prices of all the components incorporated in the PC) increased by the agent's desired profit.

When there is a very high customer demand for PCs agents usually do not send offers for all the RFQs received. After analyzing the RFQs that did not get any offers, we noticed that some of them were very profitable. As a result, we decided to send offers for them but with high offer prices. In this case the agent uses the *handleCustomerRFQsHighDemand* algorithm. It is a "greedy" algorithm since the offer prices for the PCs are always just slightly under the customer's reserve price. This algorithm is also used at times when the agent's factory does not have much free capacity for the next few days. This happens when the agent receives a lot of orders. By sending offers with very high offer prices, the agent will not get many orders. This prevents the situation in which the agent has to pay penalties because he cannot deliver all the ordered PCs. At the same time, a high profit is achieved with the few orders the agent wins.

For the last ten days of the game the agent uses the *handleCustomerRFQsEndGame* algorithm. Unlike the other two algorithms, it only sorts RFQs in increasing order of their corresponding penalties. The main purpose of this algorithm is to sell out the whole inventory in the warehouse so the profit that the agent adds to the basic PC price is minimal.

All the selling algorithms implement a mechanism used to preventing late deliveries and keep the agent from paying penalties. Each day, the agent monitors its obligations to customers by calculating the number of factory cycles needed to fulfill its existing orders for each delivery date. After analyzing these numbers, the agent calculates the earliest delivery date for sending new PC offers. This way the agent is prevented from sending offers that cannot be delivered by the requested delivery date.

3.2.2 Calculating Component Prices

The basic PC price is calculated by summing the average prices of each component incorporated in the PC. Each day the agent calculates the average component price for each component type that is in its warehouse. If some component type is not used in PC production for several days in a row, which usually means that it was purchased at a high price which is no longer concurrent on the market. As a result, the agent puts a discount on it. This way the agent prevents a further blockade of selling PCs which contain the expensive component. The discount grows as the period of component inactivity is longer.

The agent also gives a discount on components at the end of the game. A discount for the certain component is approved if the current supply of that component is higher than the calculated optimal for that day. The component supplies are checked on the following days: 170, 180, 190, 200, 210 and 215. The purpose of giving discounts at the end of the game is to sell out the components that are still in the warehouse. The leftover components represent a direct loss of money since they were paid upon delivery.

3.2.3 Calculating the Profit Margin

Some of the parameters used to calculate the profit margin are listed below.

– The *acceptedOfferPercentage* describes how many of the agent's offers resulted in customer orders,
– The *ourPriceToHigh* and *ourPriceToLow* parameters are set depending on the ratio of the agent's offer prices and the offer prices of opposing agents. If the agent's offer prices are mostly higher than the offer prices of other agents, then the *ourPriceToHigh* flag is set. In case the agent's prices are slightly lower, the *ourPriceToLow* flag is set,
– The *cyclesReservedForActiveOrders* parameter measures the number of factory cycles needed to produce all the currently active orders,
– The *factoryUtilizationAverage* describes the average factory utilization in the last week,
– The *profitDayFactor* is modified depending on the current customer PC demand. If customer demand rises, the *profitDayFactor* increases.

If the *acceptedOfferPercentage* and *cyclesReservedForActiveOrders* parameters are decreasing, the profit margin also decreases and vice-versa. After this, the profit margin is increased if the *ourPriceToLow* flag is active, or decreased if the *ourPriceToHigh* flag is active. The profit margin calculation is then corrected depending on the due date listed in the offer, current customer PC demand and the current factory utilization. The sooner the due date, the higher the profit. Finally, if there is low demand in the market segment the PC belongs to, the profit is decreased.

3.3 Managing Daily Assembly Activities and Shipping Completed Orders to Customers

Innitially, the CrocodileAgent produced PCs only after receiving customer orders, i.e. the PCs were not manufactured in advance. Later, we added the possibility of producing PCs even if nobody ordered them. Due to the stochastic nature of the TAC SCM game customer demand varies during the game. If the agent does not produce PCs and there is low demand on the PC market, a large part of the agent's factory capacities stay unutilized. If the agent produces PC stocks during a period of low PC demand, its factory will be utilized and everything will be prepared for a period of high PC demand. In some cases, the agent can produce more PCs than can be sold by the end of the game since it can not be know for sure what the future demand is going to be. We tried to lower this risk by introducing quantity limits which represent the maximum number of PCs which can be available in stock. These limits are modified during of the game. As the end of the game approaches, they are lowered accordingly. The limits also differ for each PC market segment. For example, if we predict that the demand for Mid Range PCs will to be high, we increase the limits for the PCs in that range. The list of active orders is sorted in chronological order of the delivery dates and then a simplified algorithm for PC production and delivery to customers is executed. The algorithm runs as follows:

– If there are enough PCs in the warehouse to fulfill the order, they are reserved and added to the delivery schedule. If there are not enough PCs, but there are enough components to produce the requested PCs, the components are reserved and the agent tries to add them to the production schedule. The production demand will be successfully fulfilled only if there is enough free factory capacity available for the next day,
– After analyzing all the active orders, the agent makes plans for creating PC stocks. In order to create the PC stocks the agent checks the amount of free capacity available for the following day, are there enough components to produce the PCs and which PC types can be produced without creating a larger stock.

4 TAC SCM 2005 Competition

The 2005 TAC SCM competition was divided into three parts: qualifying round held from June 13th-24th, seeding rounds held from July 11th -22nd and final round held from August 1st-3rd. There were 32 teams competing in the qualifying and 25 teams in the seeding rounds. 24 teams competed in the finals.

During the qualifying round, the CrocodileAgent was in an intermediate phase of development. In the first week, the agent played with a slightly modified component procurement algorithm with respect that used by the KrokodilAgent. After perceiving that last year's algorithm was no longer good enough, the team started improving it. Although a slight improvement was already visible in the second week, the algorithm was finally finished in time for the seeding rounds with only minor modifications were added for the final rounds. The CrocodileAgent's final score after the qualifying round was 13.79M which was enough to place the agent in 16th place.

After the qualifying round, a considerable number of improvements were made on the CrocodileAgent. We completed the supply side of the agent and introduced the possibility of producing PCs for stocking in case the factory had a lot of free capacity available. Finally, the agent's customer side was modified with the functionalities described earlier. At the beginning of the seeding rounds, the agent had some problems with penalties. That was successfully solved by implementing a mechanism for preventing the agent from sending offers which he would not be able to fulfill. The CrocodileAgent ended the seeding rounds at 12th place with an average score of 8.48M.

Before the quarterfinals, final changes on the agent were made. While analyzing the games played in the seeding rounds, we noticed that the agent was not functioning the way we wanted it to in the low demand games. In case of low customer demand, the agent had long periods of very low factory utilization and was practically inactive. PCs were not sold since the offer prices were too high compared to those of other agents. In prior competitions, the agents were very focused on getting enough components due to the problem mentioned in Section 3.1 that caused by day-0 component orders. Thus, they paid less attention to customers. For the TAC SCM 2005, the problem with suppliers was solved by the new supply procurement model. As a result, the agents had enough components to produce PCs and could therefore turn to customers and improve their selling tactics. The CrocodileAgent took 4th place in the quarterfinals with an average score of 11.64M and ended its participation in 2005 TAC SCM.

5 Controlled Experiments

To evaluate the performance of our agent, we held a competition with some of the best agents from the TAC SCM 2005 competition. The chosen opponents were: TacTex [15], Mertacor [9], DeepMaize [16], MinneTAC and PhantAgent. All the agents were downloaded from the official TAC web page (http://www.sics.se/tac). The competition was held in our laboratory and consisted of 20 games. The final ranking is shown in Table 1. After the competition finished, we conducted a detailed analysis of the games played. The majority of the analysis was done with the CMieux Analysis and Instrumentation Toolkit for TAC SCM [17].

The first task was to analyze component purchases. After gathering information regarding the prices the agents paid for each type of component and the quantities they purchased, we calculated the average prices.

The average prices the agents paid for the components are shown in Figure 3. Components marked with ID 1xx are CPUs, 2xx are motherboards, 3xx are memory and 4xx are hard disks.

Table 1. Competition results at server `mobility3.labs.tel.fer.hr`

Position	Agent	Score	Games Played	Zero Games
1	TacTex	5 638 295.70	20	0
2	DeepMaize	4 581 805.20	20	0
3	Mertacor	1 364 353.85	20	0
4	PhantAgent	976 780.00	20	0
5	MinneTAC	-426 722.20	20	0
6	CrocodileAgent	-1 243 212.45	20	0

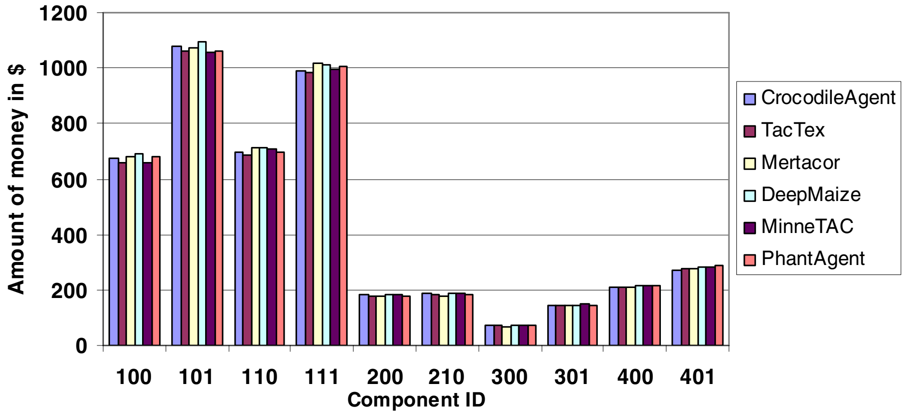


Fig. 3. Average component prices

It is very important to purchase cheap CPUs since the price of a CPU accounts for more than 50 % of the PC price. We can see from Figure 3 that DeepMaize paid the highest prices for CPUs while TacTex bought some of the cheapest CPUs. The CrocodileAgent bought the third cheapest CPUs. This is interesting since the CrocodileAgent does not have a sophisticated algorithm with supplier capacity estimations and customer demand predictions to determine component prices and required quantities. The situation is different with other components. Also, we can see that Mertacor bought the cheapest motherboards and memory, while the CrocodileAgent bought the cheapest hard disks. MinneTAC's purchase algorithm could also be improved since it bought the most expensive motherboards, memory and hard disks.

Furthermore, we analyzed the quantity of components bought during the game. From Figure 4, we can see that TacTex, DeepMaize and the CrocodileAgent bought larger amounts then the other three agents. We can also see the average minimal number of components that stayed in the agents' warehouses. The maximum number of PCs that can be assembled is equal to the minimal amount of a certain component type since all four component types are needed to produce a PC.

The leftover components represent a direct loss of money since they were paid for upon delivery. The best results regarding leftover component management were obtained by the PhantAgent and DeepMaize. The majority of their leftover components were memory. This is desirable since memory is the cheapest component type. The Mertacor agent also had a relatively small number of leftover components, but they were mostly CPUs. This is not as good since CPUs are at least ten times more expensive than memory. All three agents had between 500 and 800 leftover components.

Two agents had more than 2600 leftover components on average. These agents were TacTex and MinneTAC. The difference is that half of TacTex's leftover components were CPUs, while half of MinneTAC's leftover components were memory. The CrocodileAgent was in the middle with an average of 1250 leftover components.

This is a direct consequence of the ordering algorithm that does not compare the total number of ordered components of each component type. The total quantity of sold PCs is directly linked to the quantities of components bought and thus will not be discussed in detail.

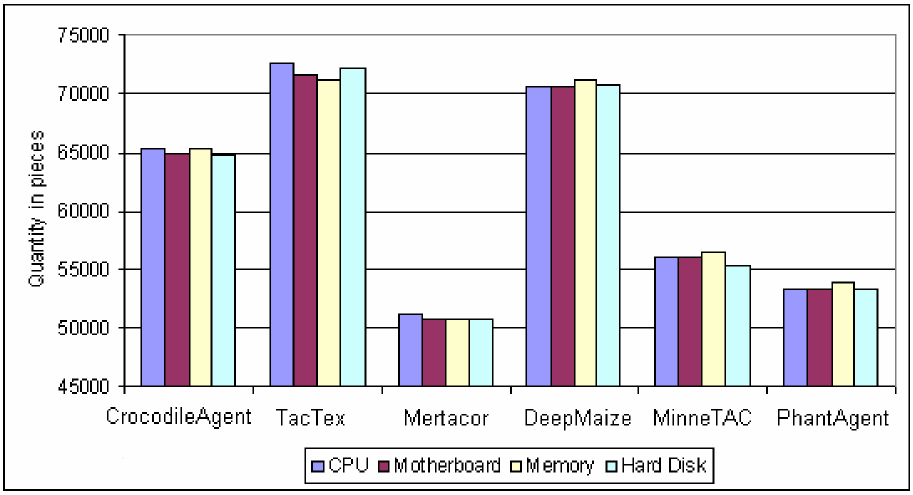


Fig. 4. Total quantity of bought components

The quantities of sold PCs are shown in Figure 5. We can see that all the agents sold more Mid Range PCs than any other type. The reason is very simple. Namely, out of the 16 PC types available, there are 6 PCs classified as Mid Range PCs while the remaining 10 are equally divided into Low and High Range PCs. We can also see that all the agents tried to sell all PCs types but, surprisingly, only MinneTAC sold more High Range than the Low Range PCs.

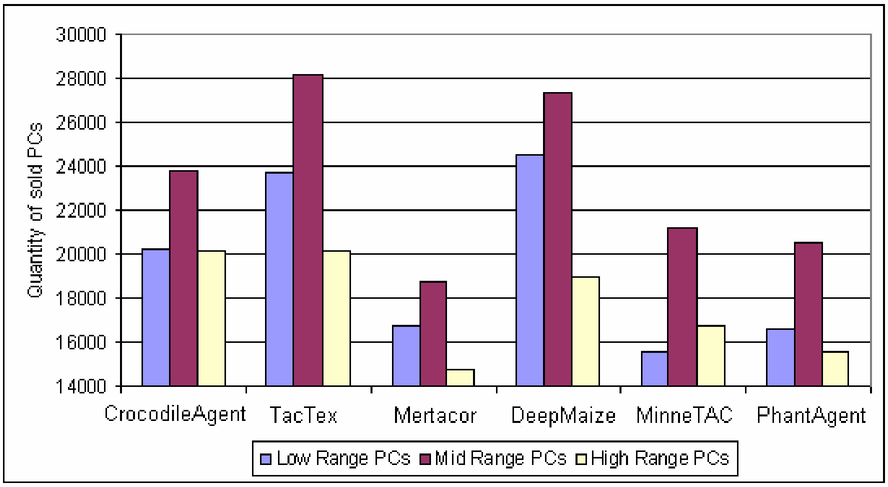


Fig. 5. Total quantity of sold PCs

The CrocodileAgent sold an equal number of Low and High Range PCs, while the other agents sold significantly larger quantities of Low Range than High Range PCs. There are several explanations for this. First of all, the demand for Low Range PCs was slightly higher than the demand for High Range PCs. Also, the penalties for Low Range PCs were lower and thus posed a lower potential risk if the order could not be delivered. The agents assumed that other agents were going to focus on selling High and Mid Range PCs since they are supposed to bring higher profits.

If we look at the average PC selling prices in Figure 6, we can see the obvious reason for the CrocodileAgent placing last in this competition. Namely, it sold cheapest PCs. The reason for this lies in the logic of the PC selling algorithm, particularly in the way the agent determines his desired profit. Further proof that the profit determining mechanism did not function well is the fact the CrocodileAgent had the highest average price of its loosing offers. If we look at the selling algorithm in Section 3.2, it seems logical but the main disadvantage of the algorithm is its lack of a winning price prediction mechanism that was used by the other agents. DeepMaize certainly has one of the best prediction mechanisms and thus achieved the highest average PC selling prices for Low and Mid Range PCs and the second highest average PC selling price for High Range PCs. Another agent with a successful selling algorithm is the MinneTAC agent which achieved the highest average PC selling price for High Range PCs. Here, we can see MinneTAC's orientation towards the High Range PC market.

To conclude, we can say that TacTex won the competition due to its highly efficient component procurement algorithm and a good PC selling algorithm. The large number of components bought and PCs sold only contributed to the victory.

DeepMaize lost first place due to its very high component purchase prices, especially CPUs. Mertacor performed best between the three agents which sell smaller amounts of components. It bought rather expensive CPUs but some of the cheapest motherboards, memory and hard disks. Its PC selling prices were average and that was enough for it to place third in the competition.

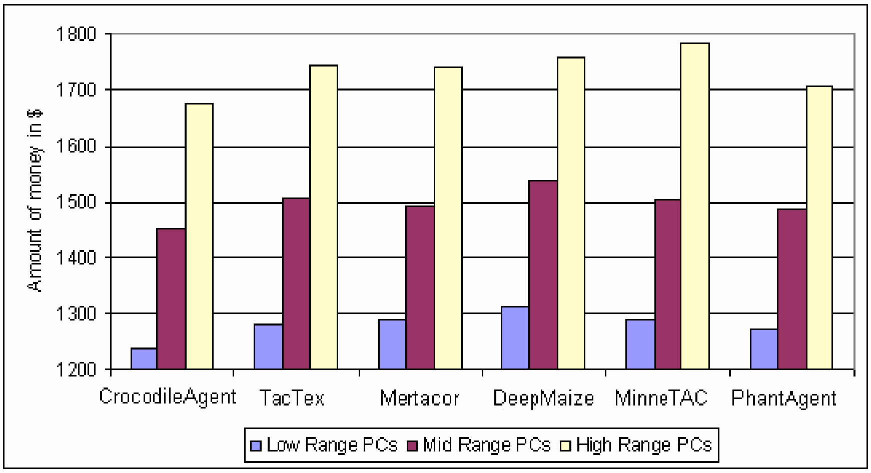


Fig. 6. Average PC selling prices

Still, the difference between the first two agents and the Mertacor agent is significant. The reason lies in the big difference in the number of PCs sold and the smaller profit made by Mertacor per PC.

The PhantAgent bought rather cheap components but also sold some of the cheapest PCs. When combined with the small amount of PCs sold, the final score of the PhantAgent put it in fourth place. The MinneTAC agent performed best in selling large quantities of High Range PCs at very high prices. In spite of that, MinneTAC finished in fifth place. Some of the reasons include very high component purchase prices and a small total number of PCs sold.

6 Conclusions and Future Work

In this paper, presented is the CrocodileAgent which is a trading agent that participated in TAC SCM 2005. After a short game description, we listed the basic agent tasks and explained how they were implemented in the CrocodileAgent. We described its strategy of day-0 component procurement combined with ordering components during the game. We then presented an algorithm for selling PCs where special attention is paid to determining the desired profit. The description of CrocodileAgent's algorithms was concluded with a list of improvements that were made in the production and delivery mechanisms. We added the option of producing stocks of PCs to the already existing mechanism of producing PCs after customer orders arrive.

We briefly presented the results of the CrocodileAgent in the TAC SCM 2005. In order to improve the functionalities of the agent, we held a competition with some of the best agents in TAC SCM 2005. We figured that this was a good way to determine the agent's soft spots. A thorough analysis of the competition was conducted. The results were a little discouraging since the CrocodileAgent placed last, but a lot was learned. The main reason for the CrocodileAgent's results lies in its reactive algorithm for selling PCs. This algorithm does not predict the fluctuation of prices on the PC market. Instead, it only reacts to the current state of the PC market and regulates its desired profit per PC. The component purchase algorithm functions quite well, but there is always room for improvement, particularly regarding motherboard and memory components.

The changes of the component purchase rules introduced in TAC SCM 2005 required a lot of work on the component purchase algorithm. Since there are no major changes in TAC SCM 2006, the CrocodileAgent represents a good basis for further development. Special attention needs to be dedicated to the PC selling algorithm with an emphasis on customer demand prediction and the prediction of winning PC prices. At this time, we are conducting some experiments which introduce fuzzy logic into the agent to help improve some of its functionalities.

References

1. Podobnik, V., Trzec, K., and Jezic, G.: An Auction-Based Semantic Service Discovery Model for E-Commerce Applications. Lecture Notes in Computer Science, Vol. 4277. Springer-Verlag, Berlin Heidelberg New York (2006). 97-106
2. Carlson, B.: The Digital Economy: What is New and What is Not?. Structural Change and Economic Dynamics, Vol. 15 (3). Elsevier, North-Holland (2004). 245-264

3. Bradshaw, J.M.: Software Agents. MIT Press, Cambridge, Massachusetts, USA (1997)
4. Podobnik, V., Petric, A., Jezic, G.: The CrocodileAgent: Research for Efficient Agent-Based Cross-Enterprise Processes. Lecture Notes in Computer Science, Vol. 4277. Springer-Verlag, Berlin Heidelberg New York (2006). 752-762
5. Kusek, M., Lovrek, I. Sinkovic, V.: Agent Team Coordination in the Mobile Agent Network. Lecture Notes in Artificial Intelligence, Subseries of Lecture Notes in Computer Science, Vol. 3053. Springer-Verlag, Berlin Heidelberg New York (2005). 240-246
6. Lin, F., Sung, Y., Lo,Y.: Effects of Trust mechanisms on Supply-Chain Performance: A Multi-Agent Simulation Study. International Journal of Electronic Commerce, Vol. 9 (4). M.E. Sharpe, New York (2005). 91-112
7. Collins, J., Arunachalam, R., Sadeh, N., Eriksson, J., Finne, N., Janson, S.: The Supply Chain Management Game for the 2005 Trading Agent Competition. http://www.sics.se/tac/tac05scmspec_v157.pdf. Date accessed: Dec 21, 2006.
8. Eriksson, J., Finne, N., Janson, S.: Evolution of a Supply Chain Management Game for the Trading Agent Competition. AI Communications, Vol. 19 (1). IOS Press, Amsterdam (2006). 1-12
9. Kontogounis, I., Chatzidimitriou, K.C., Symeonidis, A.L., Mitkas, P.A.: A Robust Agent Design for Dynamic SCM Environments. In Proceedings of the 4th Hellenic Joint Conference on Artificial Intelligence (SETN), Heraklion, Greece, 2006. 127-136
10. Benish, M., Sardinha, A., Andrews, J., Sadeh, N.: CMieux: Adaptive Strategies for Competitive Supply Chain Trading. In Proceedings of the 8th Int. Conference on Electronic Commerce (ICEC), Fredericton, Canada, 2006. 1-10
11. Jordan, P.R., Kiekintveld, C., Miller, J., Wellman, M.P.: Market Efficiency, Sales Competition, and the Bullwhip Effect in the TAC SCM Tournaments. In Proceedings of the AAMAS Joint International Workshop on the Trading Agent Design and Analysis and Agent Mediated Electronic Commerce (TADA/AMEC) Hakodate, Japan, 2006. 99-111
12. Ketter, W., Collins, J., Gini, M., Gupta, A., Shrater, P.: Identifying and Forecasting Economic Regimes in TAC SCM. In Proceedings of the IJCAI Workshop on Trading Agent Design and Analysis (TADA), Edinburgh, UK, 2005. 53-60
13. Petric, A., Podobnik, V., Jezic, G.: The CrocodileAgent: Analysis and Comparison with Other TAC SCM 2005 Agents. In Proceedings of the AAMAS Joint International Workshop on the Trading Agent Design and Analysis and Agent Mediated Electronic Commerce (TADA/AMEC) Hakodate, Japan, 2006. 202-205
14. Petric, A., Jurasovic, K.: KrokodilAgent: A Supply Chain Management Agent. In Proceedings of the 8th International Conference on Telecommunications (ConTEL), Zagreb, Croatia, 2005. 297-302
15. Pardoe, D., Stone, P.: Bidding for Customer Orders in TAC SCM: A Learning Approach. In Proceedings of the AAMAS International Workshop on Trading Agent Design and Analysis (TADA), New York, USA, 2004.
16. Wellman, M.P., Estelle, J., Singh, S., Vorobeychik, Y, Kiekintveld, C., Soni, V.: Strategic Interactions in a Supply Chain Game. Computational Intelligence, Vol. 21 (1). Blackwell Publishing, Oxford, UK (2005). 1-26
17. Benisch, M., et. al.: CMieux Analysis and Instrumentation Toolkit for TAC SCM. Carnegie Mellon University School of Computer Science TR CMU-ISRI-05-127 (2005)

A Fuzzy Constraint Based Model for Automated Purchase Negotiations[⋆]

Miguel A. López-Carmona and Juan R. Velasco

Department of Automática, University of Alcalá
Edificio Politécnico, Ctra. NII, km. 33.600
28871 Alcalá de Henares (Madrid), Spain
{miguellop,juanra}@aut.uah.es

Abstract. This paper presents a fuzzy constraint based model for bilateral multi-attribute agent purchase negotiations in competitive trading environments. Fuzzy constraints are used to capture requirements and to express proposals. The proposed interaction protocol is a dialogue game protocol where argumentation is used as a key mechanism to improve agreements in contrast to other fuzzy constraint based models which are limited to quantitative offers and counter-offers. A set of locutions and decision mechanisms which fire them are fully specified, so that each agent may decide its degree of cooperation and its degree of expressiveness, which in turn may have effects on the quality of the agreement. The notions of similarity and expected valuations of products are used in order to design efficient decision mechanisms. An example of a purchase scenario and a summary of statistical tests are presented to demonstrate the proposed model.

1 Introduction

Many approaches to automated negotiation in e-business have been proposed, covering different areas such as game theory, evolutionary computation and distributed artificial intelligence. In [1] are identified the main parameters on which any automatic negotiation depends, and a classification schema is used to categorize the different negotiation scenarios. Fuzzy constraints have been used in several models and approaches to multi-attribute agent negotiation [2,3,4]. As Luo et al. declare in [2], there are several reasons which make very convenient the use of fuzzy constraints as the core of a negotiation model: it is an efficient way of capturing requirements [5]; fuzzy constraints are capable of representing trade-offs between the different possible values for attributes; and, using constraints to express offers in turns means that the solution space can be explored in a given exchange and so means that the search for an agreement is more efficient. Luo et al. [2] developed a fuzzy constraint based model for bilateral,

⋆ This work has been supported by the Spanish Ministry of Science and Technology grant MCYT-TIC2003-09192-C11-05, and by the University of Alcalá grant UAH-PI-2003/001.

multi-issue negotiations in semi-competitive environments. In their model the buyer's preferences are captured through prioritized fuzzy constraints [6]. The problem with this approach is that the ability of the seller agent to express its needs is limited, and so the joint decision process is not balanced. In [3] a general problem-solving framework for modelling multi-issue multilateral agent negotiation using fuzzy constraints is presented. In this model all parties model their preferences as fuzzy constraints, and a meta-strategy based on a concession and a trade-off strategies is presented to evaluate existing alternatives. However, the proposals are not expressed as constraints but as concrete offers, and preferences are not explicitly propagated. Faratin et al. [7] developed a similar model for performing trade-offs in automated negotiations but where fuzzy constraints are not used.

Our research focuses on multi-attribute bilateral negotiation using fuzzy constraints in competitive environments. We present a negotiation model where the buyer's demands are expressed as fuzzy constraints and the sellers have a set of products which are characterized as a vector of attributes and a profit. Our approach lets the buyer agent value the degree of importance that each submitted constraint has, and lets the seller agent inform how constraints should be relaxed, so what we do is to argue proposals or offers by means of qualifications.[1] We propose a general framework for these mechanisms which is based on a measure of similarity between each product in the catalogue and the constraints received from the buyer agent, the beliefs about the valuation the buyer agent gives to the products, and the associated profit. We will show how the possibility of reaching an agreement which is more favorable for both parties is increased.

The remainder of this paper is organized as follows. Section 2 recalls the most relevant concepts our research is based on, and presents some preliminaries. Section 3 details the interaction protocol, while Section 4 presents the decision mechanisms and Section 5 describes the transition rules. Section 6 presents an example of purchase scenario and an experimental analysis. Finally, Section 7 presents the conclusions and future work.

2 Preliminaries

2.1 The Fuzzy Constraint Satisfaction Problems (FCSPs) Framework

In this subsection we recall the necessary concepts related to FCSPs [2,11,12]. A fuzzy constraint satisfaction problem (FCSP) is a 3-tuple $\left(X, D, C^{\mathrm{f}}\right)$, where $X = \{x_i | = 1, ..., n\}$ is a finite set of variables, $D = \{d_i | = 1, ..., n\}$ is the set of domains, and $C^{\mathrm{f}} = \{R_i^{\mathrm{f}} | R_i^{\mathrm{f}} \subseteq \Pi_{x_j \in var(R_i^{\mathrm{f}})} d_j, i = 1, ..., m\}$ is a set of fuzzy constraints, where $var(R_i^{\mathrm{f}})$ denotes the set of variables associated with each constraint R_i^{f}. The function that indicates how well a given constraint is satisfied is the *satisfaction degree function* $\mu_{R_i^{\mathrm{f}}} : \left(\Pi_{x_j \in var(R_i^{\mathrm{f}})} d_j\right) \rightarrow [0, 1], i = 1, ..., m,$

[1] Argumentation has been recognized as a key tool in negotiation protocols [8,9,10].

where 1 indicates completely satisfied and 0 indicates not satisfied at all. Using a cut-set technique a fuzzy constraint can induce a *crisp constraint*. Given the *cut level* $\sigma \in [0,1]$, the induced crisp constraint of a fuzzy constraint R^{f} is defined as R^{c}. Finally, the *overall (or global) satisfaction degree* of a potential solution v_X is $\alpha(v_X) = \oplus \left\{ \mu_{R^{\mathrm{f}}}(v_X) | R^{\mathrm{f}} \in C^{\mathrm{f}} \right\}$, where $\oplus$ is an aggregation[2] from $[0,1]^m$ to $[0,1]$.

2.2 Domain Knowledge of the Buyer Agent

Let $\mathcal{B}_{\mathrm{req}} = (X, D, C^{\mathrm{f}}, N_{\mathrm{b}})$ be the requirement model of the buyer agent, which is defined as a FCSP $\mathcal{F} = (X, D, C^{\mathrm{f}})$ which expresses the requirements on the attributes; and a negotiation profile $N_{\mathrm{b}} = \{\xi, \eta\}$, where $\xi \in \{0,1\}$ controls the use of purchase requirement valuations, and $\eta \in [0,1]$ modulates the buyer's attitude regarding a relax requirement.

Definition 1. (Purchase Requirement) *Let $R_i^{\mathrm{c}(\sigma_i)}$ be a crisp constraint induced from R_i^{f} at a cut level σ_i, a purchase requirement is defined as $\lambda_{\mathcal{B}_{\mathrm{req}}} = \bigcap \left\{ R_i^{\mathrm{c}(\sigma_i)} \right\}$. A purchase requirement may be formed by a subset of the fuzzy constraints in C^{f}, where $\mathrm{cons}(\lambda_{\mathcal{B}_{\mathrm{req}}})$ denotes the set of constraints included in a $\lambda_{\mathcal{B}_{\mathrm{req}}}$.*

Definition 2. (Potential Overall Satisfaction Degree) *Given a purchase requirement, the potential overall satisfaction degree (posd) the buyer may get if the requirement is met is $\alpha^{\lambda_{\mathcal{B}_{\mathrm{req}}}} = \oplus \{\sigma_i | i = 1, ..., m)\}$, where σ_i represents the cut level applied to constraint R_i^{f}, and the cut level applied to constraints not included in the purchase requirement takes value 1.*

One of the key decision mechanisms of the buyer agent is related to defining a strategy to generate a purchase requirement.[3]

Definition 3. (Concession Strategy) *Given a purchase requirement $\lambda^k_{\mathcal{B}_{\mathrm{req}}}$ submitted at instant k, a concession strategy may be defined as a mechanism which generates a new $\lambda^{k+1}_{\mathcal{B}_{\mathrm{req}}}$ such that $\alpha^{\lambda^{k+1}_{\mathcal{B}_{\mathrm{req}}}} < \alpha^{\lambda^{k}_{\mathcal{B}_{\mathrm{req}}}}$, where $\alpha^{\lambda^{k}_{\mathcal{B}_{\mathrm{req}}}} - \alpha^{\lambda^{k+1}_{\mathcal{B}_{\mathrm{req}}}}$ is minimized.*

Definition 4. (Trade-off Strategy) *Given a purchase requirement $\lambda^k_{\mathcal{B}_{\mathrm{req}}}$ submitted at instant k, a trade-off strategy may be defined as a mechanism which generates a new requirement $\lambda^{k+1}_{\mathcal{B}_{\mathrm{req}}}$ such that $\alpha^{\lambda^{k+1}_{\mathcal{B}_{\mathrm{req}}}} \geq \alpha^{\lambda^{k}_{\mathcal{B}_{\mathrm{req}}}}$* and *$\lambda^{k+1}_{\mathcal{B}_{\mathrm{req}}} \neq \lambda^{j}_{\mathcal{B}_{\mathrm{req}}} | \forall j < k+1$.*

Definition 5. (Purchase Requirement Valuation) *Given a purchase requirement $\lambda^k_{\mathcal{B}_{\mathrm{req}}}$, a purchase requirement valuation is defined as a set $v_{\mathcal{B}_{\mathrm{req}}} = \{v_i | v_i \in [0,1], i \in \mathrm{cons}(\lambda_{\mathcal{B}_{\mathrm{req}}})\}$, where v_i is the degree of importance that the constraint i has for the buyer agent.*

[2] The choice of an appropriate aggregation operator depends on the context of the negotiation and is directly related to the negotiators attitudes.

[3] This strategy has been commonly defined as a meta-strategy based on concession and trade-off strategies.

2.3 Domain Knowledge of the Seller Agent

Let $\mathcal{S}_{\text{req}} = (S, N_{\text{s}})$ be the requirement model of the seller agent, which is defined by the set of products the seller holds, $S = \{s_j | s_j = (p_j, u_j), p_j = (a_{j1}, ..., a_{jn}), 0 \leq j \leq k\}$, where p_j is the vector of attributes, u_j is the profit, and k is the total number of products in the catalogue; and a negotiation profile $N_{\text{s}} = \{\psi, \beta, \Delta^k\}$, where $\psi \in \{0, 1\}$ controls whether the seller agent expresses its preferences for a specific relaxation of the previous buyer's demands, $\beta \in [0, 1]$ modulates the seller's attitude regarding a purchase requirement, and $\Delta^k = \{(\delta_i^k, \gamma_i^k), i = 1, ..., m\}$ defines a dynamic set of beliefs, where δ_i^k represents the *beliefs* about the relaxation strategy used by the buyer agent at instant k and for constraint i, and γ_i^k is the *degree of certainty*.[4]

Definition 6. (Sale Offer) *A sale offer is defined by a product p_j.*

Definition 7. (Relax Requirement) *A relax requirement is defined as a set $\rho_{\mathcal{B}_{\text{req}}} = \{r_i | r_i \in [0, 1], i \in \text{cons}(\lambda_{\mathcal{B}_{\text{req}}})\}$, where r_i is the preference for constraint i to be relaxed.*

3 A Purchase Negotiation Dialogue

In [9] McBurney et al. introduced a framework for automating purchase negotiations.[5] Our purchase negotiation dialogue is built upon this framework, and is defined as a four stage sequence: *open dialogue (L1-2), negotiate (L3-8), confirm (L9-10) and close dialogue (L11).*

L1: open_dialogue$(P_{\text{b}}, P_{\text{s}}, \theta)$ P_{b} suggests the opening of a purchase dialogue to a seller participant P_{s} on product category θ. P_{s} agent wishing to participate must respond with **enter_dialogue(.)**.

L2: enter_dialogue$(P_{\text{s}}, P_{\text{b}}, \theta)$ P_{s} indicates a willingness to join a purchase dialogue with participant P_{b}. Within the dialogue, a participant P_{b} must have uttered the locution **open_dialogue(.)**.

L3: willing_to_sell$(P_{\text{s}}, P_{\text{b}}, p_j)$ P_{s} indicates to the buyer P_{b} a willingness to sell a product p_j. A buyer P_{b} must have uttered a **desire_to_buy(.)** or a **prefer_to_buy(.)** locution.

L4: desire_to_buy$(P_{\text{b}}, P_{\text{s}}, \lambda_{\mathcal{B}_{\text{req}}})$ P_{b}, speaking to the seller P_{s}, requests to purchase a product which satisfies the purchase requirement $\lambda_{\mathcal{B}_{\text{req}}}$.

L5: prefer_to_sell$(P_{\text{s}}, P_{\text{b}}, \lambda_{\mathcal{B}_{\text{req}}}, \rho_{\mathcal{B}_{\text{req}}})$ P_{s}, speaking to the buyer P_{b}, requests to relax the purchase requirement $\lambda_{\mathcal{B}_{\text{req}}}$, and expresses which constraints are preferred to be relaxed, by means of the relax requirement $\rho_{\mathcal{B}_{\text{req}}}$.

[4] The degree of certainty does not have a probabilistic interpretation.

[5] Their work is concerned with the locutions and interaction protocols, but less concerned with the decision mechanisms. Nevertheless, this framework provides a solid characterization of complex dialogues.

L6: prefer_to_buy$(P_\mathrm{b}, P_\mathrm{s}, \lambda^k_{\mathcal{B}_\mathrm{req}}, \upsilon_{\mathcal{B}_\mathrm{req}})$ P_b, speaking to the seller P_s, requests to purchase a product which satisfies the purchase requirement $\lambda^k_{\mathcal{B}_\mathrm{req}}$, and expresses its preferences for the different constraints by means of the purchase requirement valuation $\upsilon_{\mathcal{B}_\mathrm{req}}$.

L7: refuse_to_buy$(P_\mathrm{b}, P_\mathrm{s}, p_j)$ The buyer agent expresses a refusal to purchase a product p_j. This locution cannot be uttered following a valid utterance of **agree_to_buy(.)**.

L8: refuse_to_sell$(P_\mathrm{s}, P_\mathrm{b}, p_j|\lambda_{\mathcal{B}_\mathrm{req}})$ The seller agent expresses a refusal to sell a product p_j, or it expresses a refusal to sell products which satisfy the purchase requirement $\lambda_{\mathcal{B}_\mathrm{req}}$. This locution cannot be uttered following a valid utterance of **agree_to_sell(.)**.

L9: agree_to_buy$(P_\mathrm{b}, P_\mathrm{s}, p_j)$ Buyer agent P_b speaking to P_s commits to buy the product p_j. A locution of the form **willing_to_sell(.)** must have been uttered.

L10: agree_to_sell$(P_\mathrm{s}, P_\mathrm{b}, p_j)$ Seller agent P_s speaking to buyer agent P_b commits to sell the product p_j. A locution of the form **agree_to_buy(.)** must have been uttered.

L11: withdraw_dialogue$(P_\mathrm{x}, P_\mathrm{y}, \theta)$ For P_x and P_y participants with different roles (i.e. sellers and buyers). P_x announces to agent P_y the withdrawal from the dialogue.

Next step is to specify the mechanisms which will invoke particular locutions in the course of a dialogue.

4 The Decision Mechanisms

The following mechanisms are grouped according to the type of participants: Buyers (**B**) or Sellers (**S**). Now, we present the buyer's mechanisms.

B1: Recognize Need enables the buyer agent to recognize a need for a purchase. A user who wants to buy a product initiates a purchase dialogue by means of a user interface. **Outputs:** have_need(θ).

B2: Generate Purchase Requirement enables the buyer agent to generate a new purchase requirement to include in the *desire_to_buy()* or the *prefer_to_buy()* locutions. **Outputs:** empty set $\emptyset$, $\lambda^k_{\mathcal{B}_\mathrm{req}}$.

There are two basic sub-mechanisms to generate a new purchase requirement: **(a)** *add* a new constraint, when the buyer agent is preparing the first purchase requirement, or when a sale offer received from the seller agent cannot be accepted (and new constraints exist); and **(b)**, *relax* the current purchase requirement applying when possible the trade-off strategy, and otherwise the concession one.[6] If the buyer agent receives the *prefer_to_sell()* locution instead

[6] With this approach, the loss of *posd* is minimized. There is no reason for the buyer agent to deviate from this strategy when the only available criterium is the local utility.

of the *refuse_to_sell()* locution, a variation to this strategy can be applied which takes into account the relax requirement $\rho_{\mathcal{B}_{\text{req}}}$. We call this, the *cooperative purchase requirement relaxation strategy* which is defined below. An empty set $\emptyset$ is generated if sub-mechanisms **(a)** or **(b)** cannot be used.

Definition 8. (Cooperative Purchase Requirement Relaxation Strategy) *Given a relax requirement* $\rho^k_{\mathcal{B}_{\text{req}}}$ *as a response to a purchase requirement* $\lambda^k_{\mathcal{B}_{\text{req}}}$*, a cooperative purchase requirement relaxation strategy is defined as a mechanism which generates a new purchase requirement* $\lambda^{k+1}_{\mathcal{B}_{\text{req}}\text{cooperative}}$ *as a function of* $\rho^k_{\mathcal{B}_{\text{req}}}$ *and* $\lambda^k_{\mathcal{B}_{\text{req}}}$*, which must integrate the demands of the seller agent.*

The *relax requirement* acts as an argument which explains why a purchase requirement has not been accepted, so it allows seller agent to explicitly influence buyer's preferences during negotiation. Finally, we propose the following algorithm to implement a *generalized purchase requirement strategy*:

Algorithm 1: (1) Given $\lambda^k_{\mathcal{B}_{\text{req}}}$ obtain the set $\{\alpha_{R^{\text{f}}_i}^{\lambda^{k+1}_{\mathcal{B}_{\text{req}}}} | \forall i \in \text{cons}(\lambda^{k+1}_{\mathcal{B}_{\text{req}}})\}$, where $\alpha_{R_i}^{\lambda^{k+1}_{\mathcal{B}_{\text{req}}}}$ represents the *posd* if constraint R^{f}_i is relaxed. (2) Compute $\alpha^{\lambda^{k+1}_{\mathcal{B}_{\text{req}}\text{cooperative}}} = \odot(\{\alpha_{R^{\text{f}}_i}^{\lambda^{k+1}_{\mathcal{B}_{\text{req}}}} | \forall i \in \text{cons}(\lambda^{k+1}_{\mathcal{B}_{\text{req}}})\})$, where $\odot$ is an aggregation operator. This value will serve as a threshold in order to determine which constraints can be relaxed attending to *overall satisfaction degree* criteria. (3) Select from $\{\alpha_{R^{\text{f}}_i}^{\lambda^{k+1}_{\mathcal{B}_{\text{req}}}} | \forall i \in \text{cons}(\lambda^{k+1}_{\mathcal{B}_{\text{req}}})\}$ those elements such that $\alpha_{R^{\text{f}}_i}^{\lambda^{k+1}_{\mathcal{B}_{\text{req}}}} \geq \alpha^{\lambda^{k+1}_{\mathcal{B}_{\text{req}}\text{cooperative}}}$, and let $\{\lambda^{k+1}_{\mathcal{B}_{\text{req}}}\}_{\text{feasible}}$ be the set with the elements satisfying the inequality. (4) Let $\rho^k_{\mathcal{B}_{\text{req}}\text{feasible}}$ be the subset of $\rho^k_{\mathcal{B}_{\text{req}}}$ which includes the information regarding the constraints contained in $\{\lambda^{k+1}_{\mathcal{B}_{\text{req}}}\}_{\text{feasible}}$, then apply the *constraint selection function* csf which is defined as:

$$csf : (\{\lambda^{k+1}_{\mathcal{B}_{\text{req}}}\}_{\text{feasible}}, \rho^k_{\mathcal{B}_{\text{req}}\text{feasible}}, \eta) \rightarrow C^{\text{f}} \quad .$$

In our experiments[7] $csf = \arg(\max_{\{\lambda^{k+1}_{\mathcal{B}_{\text{req}}}\}_{\text{feasible}}} \alpha_{R^{\text{f}}_i}^{\lambda^{k+1}_{\mathcal{B}_{\text{req}}}} + r_i * \eta)$.

B3: Generate Purchase Requirement Valuation enables the buyer agent to obtain the valuation of a purchase requirement. **Outputs:** empty set $\emptyset$, $v_{\mathcal{B}_{\text{req}}}$.

If $\xi = 0$ this mechanism returns an empty set $\emptyset$. If $\xi = 1$ this mechanism generates the valuation of a purchase requirement. The valuation is based on the calculus of the *posd* the buyer agent gets if a constraint is relaxed. Summarizing, a high valuation implies that the buyer agent prefers not to relax the constraint.

[7] *csf* searches the constraint that when relaxed, maximizes the weighted sum of the *posd* and the *relax requirement* received from the seller agent. We can see how the η parameter modulates the buyer's attitude regarding the relax requirement.

Algorithm 2: (1) Given a purchase requirement $\lambda^{k+1}_{\mathcal{B}_{\mathrm{req}}}$ then obtain a vector with the *posd* for all the possible purchase requirements (i.e. what would happen in the next round of negotiation with the overall satisfaction degree if a constraint is relaxed): $\{\alpha^{\lambda^{k+2}_{\mathcal{B}_{\mathrm{req}}}}_{R^{\mathrm{f}}_i} | \forall i \in \mathrm{cons}(\lambda^{k+2}_{\mathcal{B}_{\mathrm{req}}})\}$. (2) Apply the *purchase requirement valuation function* $prvf : (\{\alpha^{\lambda^{k+2}_{\mathcal{B}_{\mathrm{req}}}}_{R^{\mathrm{f}}_i} | \forall i \in \mathrm{cons}(\lambda^{k+2}_{\mathcal{B}_{\mathrm{req}}})\}) \rightarrow \upsilon_{\mathcal{B}_{\mathrm{req}}}$. This function represents a general framework to obtain the valuation of a purchase requirement.[8]

B4: Consider Offers decides at a particular time whether to: **(a)** *accept* an offer if $\alpha(p_j) \geq \alpha^{\lambda^{k}_{\mathcal{B}_{\mathrm{req}}}}$; **(b)** *reject* the offer if for some reason it does not fall within a previously submitted purchase requirement; or **(c)** *generate* a new purchase requirement if the sale offer does not equal or exceed the actual *posd*. **Outputs:** accept_offer(p_j), reject_offer(p_j), generate purchase requirement(p_j),

B5: Consider Withdrawal generates the withdraw(θ) output when invoked. **Outputs:** withdraw(θ).

Now we describe the seller's mechanisms:

S1: Recognize Category assesses whether the seller has products to sell in the category. **Outputs:** enter(θ), not_to_enter(θ).

S2: Assess Purchase_Requirement assesses whether a proposed purchase requirement can be satisfied. **Outputs:** empty set $\emptyset$, $\lambda^{k}_{\mathcal{B}_{\mathrm{req}}}$, sale offer($p_j$).

Algorithm 3: (1) The seller agent searches for products which satisfy the purchase requirement. (2) If the purchase requirement cannot be satisfied and parameter $\psi = 0$, an empty set $\emptyset$ is returned. (3) If the purchase requirement cannot be satisfied and parameter $\psi = 1$, the purchase requirement $\lambda^{k}_{\mathcal{B}_{\mathrm{req}}}$ is returned. (4) From the products which satisfy the buyer's requirement, choose those with the highest utility and return the product p_j which has been a sale offer a minimum number of times.

S3: Generate Potential Sale Offers makes a selection of products which the seller agent considers as good candidates for a sale offer. **Outputs:** set S_{p} of products.

If a seller agent cannot satisfy a purchase requirement, it may encourage the buyer agent to change its proposals. A good alternative to promote a convergence in the negotiation is to express how the purchase requirement should be relaxed.[9] We plan to do this by means of a *relax requirement*. The mechanism considers the

[8] In our experiments we have defined *prvf* as a linear function, where those constraints that when relaxed generate a low *posd* are more valued than constraints which generate a higher *posd*.

[9] A seller agent is not interested in sending an explicit sale offer unless this offer matches the purchase requirement (a premature offer may imply a loss of opportunity).

first task in this process: the selection of candidates for a future sale offer. We have identified two main aspects when making this selection: the *local utility*, which depends on the u_j parameter (i.e. the profit), and the *viability*, which depends on the similarity between the p_j candidate and the purchase requirement $\lambda^k_{\mathcal{B}_{\rm req}}$, as well as on the expected buyer's valuation. A seller agent may give more or less importance to each aspect by means of the $\beta \in [0,1]$ parameter. The function $prefer$ estimates the goodness of a potential sale offer in terms of *utility* and *viability*:

$$prefer(p_j) = \beta * u_j + (1-\beta) * vi\hat{ability}(p_j, \lambda^k_{\mathcal{B}_{\rm req}}, \upsilon_{\mathcal{B}_{\rm req}}) \ ,$$

where $vi\hat{ability} = s\hat{i}m(p_j, \lambda^k_{\mathcal{B}_{\rm req}}) \diamond \hat{E_{bv}}(p_j, \lambda^k_{\mathcal{B}_{\rm req}}, \upsilon_{\mathcal{B}_{\rm req}})$. The first term in this function represents the similarity between a product and a purchase requirement, and it is defined as $1 - d\hat{i}st(p_j, \lambda^k_{\mathcal{B}_{\rm req}})$. The second term is the estimate of the product buyer's valuation. Having defined a general framework to assign a preference value to a potential sale offer, now we propose the following estimates for the $s\hat{i}m$ and $\hat{E_{\rm bv}}$ functions, and for the $\diamond$ operator. For simplicity, we limit the proposal to FCSPs where only one variable per constraint is defined, so the distance between a potential sale offer and a purchase requirement may be computed in a per-constraint basis as $d\hat{i}st(p_j, \lambda^k_{\mathcal{B}_{\rm req}}) = sqrt(\sum_{i=1}^{z} d\hat{i}st(a_i, \lambda^{k,i}_{\mathcal{B}_{\rm req}})^2/z)$, where $d\hat{i}st(a_i, \lambda^{k,i}_{\mathcal{B}_{\rm req}})$ estimates the distance from the attribute a_i to crisp constraint $R^{\rm c}_i$ in the purchase requirement $\lambda^k_{\mathcal{B}_{req}}$. We compute the distance as a function of the beliefs about the relaxation strategy δ^k_i, the degrees of certainty γ^k_i, and the distance from the nearest boundary of constraint i to the attribute a_i. A belief δ^k_i is a pair $(a^{\rm res}_i, \tau_i)$, where $a^{\rm res}_i$ defines the expected reservation value[10] for attribute a_i, and $\tau_i \in (0, \infty)$ weights the concession strategy used by the buyer agent. If the seller agent assumes a monotonic decreasing function for the relaxation of a_i, then:

$$dist(a_i, \lambda^{k,i}_{\mathcal{B}_{\rm req}}) = \begin{cases} 0 & \forall a_i > \lambda^{k,i}_{\mathcal{B}_{\rm req}} \\ 1 & \forall a_i < a^{\rm res}_i \\ (abs(\frac{a_i - \lambda^{k,i}_{\mathcal{B}_{\rm req}}}{a^{\rm res}_i - \lambda^{k,i}_{\mathcal{B}_{\rm req}}})^{1/\tau_i} - 1) * \gamma^k_i + 1 & otherwise \end{cases} .$$

The $\hat{E_{\rm bv}}$ estimate is defined in a similar way:

$$\hat{E_{\rm bv}}(p_j, \lambda^k_{\mathcal{B}_{\rm req}}, \upsilon_{\mathcal{B}_{\rm req}}) = \begin{cases} \{\sum_{i=1}^{z} \hat{E_{\rm bv}}(a_i, \lambda^{k,i}_{\mathcal{B}_{\rm req}}, v_i)\}/z & if\, \upsilon_{\mathcal{B}_{\rm req}} \neq \phi \\ 1 & otherwise \end{cases} ,$$

where $\hat{E_{\rm bv}}(a_i, \lambda^{k,i}_{\mathcal{B}_{\rm req}}, v_i) = (1 - d\hat{i}st(a_i, \lambda^{k,i}_{\mathcal{B}_{\rm req}})) * v_i$.

In our experiments we have defined $\diamond$ as the *mean* value. Now we describe the algorithm which selects the potential sale offers:

[10] This reservation value must be interpreted as an independent value which is not related to the reservation values of other attributes.

Algorithm_4: (1) Take all the products in the catalogue and compute the function $prefer$. (2) Fix a threshold[11] and select those products with a $prefer$ value exceeding this threshold. This is the set S_p.

S4: Generate Relax Requirement obtains the relax requirement $\rho_{\mathcal{B}_\mathrm{req}}$ to submit to the buyer agent. **Outputs:** $\rho_{\mathcal{B}_\mathrm{req}}$.

Given a set of potential sale offers S_p, this mechanism generates a relax requirement in order to encourage the buyer agent to buy one of the products contained in S_p. The following algorithm is proposed:

Algorithm_5: (1) For each potential sale offer see which constraints from the purchase requirement are not satisfied. (2) Generate $\rho_{\mathcal{B}_\mathrm{req}}$, where $r_i = 1$ if constraint R_i^f in $\lambda_{\mathcal{B}_\mathrm{req}}^k$ has not been satisfied for any product, and otherwise $r_i = 0$.

S5: Accept or Reject Offer decides at a particular time whether to accept or reject an *agree_to_buy()* locution made by a buyer agent. It returns accept(p_j) if the sale offer accepted by the buyer agent still exists. Otherwise returns reject(p_j). **Outputs:** accept(p_j), reject(p_j).

S6: Consider Withdrawal decides to withdraw from the dialogue. It generates the withdraw(θ) output when invoked. **Outputs:** withdraw(θ).

5 Operational Semantics

Operational semantics provides a formal linkage between the dialogue locutions and the semantic mechanisms previously defined [9]. We use the notation in [9] which defines the 3-tuple $\langle P_x, K, s\rangle$ to denote the mechanism with number K and with output s of agent P_x. The notation is self-explanatory, so we describe in detail only the first transition rules:

TR1: $\langle P_\mathrm{b}, \mathbf{B1}, \mathrm{have\_need}(\theta)\rangle \xrightarrow{\mathbf{L1}} \langle P_\mathrm{s}, \mathbf{S1}, .\rangle$ indicates that a buyer agent with a current need for a product will initiate a purchase negotiation dialogue by means of locution L1.

TR2: $\langle P_\mathrm{s}, \mathbf{S1}, \mathrm{not\_to\_enter}(\theta)\rangle \rightarrow \langle P_\mathrm{s}, \mathbf{S1}, .\rangle$ indicates that a seller agent does not wish to enter a dialogue at this time, but will review the situation after some time.

TR3: $\langle P_\mathrm{s}, \mathbf{S1}, \mathrm{enter}(\theta)\rangle \xrightarrow{\mathbf{L2}} \langle P_\mathrm{b}, \mathbf{B2}, .\rangle$ says that a seller agent which wish to enter a dialogue at this time will do so by means of an utterance of locution L2, i.e. enter_dialogue(). This utterance will lead the buyer agent to execute mechanism B2: Generate Purchase Requirement.

TR4: $\langle P_\mathrm{b}, \mathbf{B2}, \emptyset\rangle \rightarrow \langle P_\mathrm{b}, \mathbf{B5}, .\rangle$ says that when mechanism B2: Generate Purchase Requirement leads to an output of an empty set $\emptyset$ in a buyer agent, the B5:Consider Withdrawal mechanism is invoked.

[11] We have fixed this value to the maximum value of $prefer$. The threshold influences the number of elements of S_p. A high threshold implies a more selective criterium to generate relax requirements.

TR5: $\langle P_\mathrm{b}, \mathbf{B5}, \mathrm{withdraw}(\theta)\rangle \underrightarrow{\mathbf{L11}} \langle P_\mathrm{s}, \mathbf{S6}, .\rangle$
TR6: $\langle P_\mathrm{s}, \mathbf{S6}, \mathrm{withdraw}(\theta)\rangle \underrightarrow{\mathbf{L11}} \langle P_\mathrm{b}, \mathbf{B5}, .\rangle$
TR7: $\left\langle P_\mathrm{b}, \mathbf{B2}, \lambda^k_{\mathcal{B}_\mathrm{req}}\right\rangle \rightarrow \langle P_\mathrm{b}, \mathbf{B3}, .\rangle$
TR8: $\langle P_\mathrm{b}, \mathbf{B3}, \emptyset\rangle \underrightarrow{\mathbf{L4}} \langle P_\mathrm{s}, \mathbf{S2}, .\rangle$
TR9: $\left\langle P_\mathrm{b}, \mathbf{B3}, \upsilon_{\mathcal{B}_\mathrm{req}}\right\rangle \underrightarrow{\mathbf{L6}} \langle P_\mathrm{s}, \mathbf{S2}, .\rangle$
TR10: $\langle P_\mathrm{s}, \mathbf{S2}, \emptyset\rangle \underrightarrow{\mathbf{L8}} \langle P_\mathrm{b}, \mathbf{B2}, .\rangle$
TR11: $\langle P_\mathrm{s}, \mathbf{S2}, \mathrm{saleoffer}(p_j)\rangle \underrightarrow{\mathbf{L3}} \langle P_\mathrm{b}, \mathbf{B4}, .\rangle$
TR12: $\left\langle P_\mathrm{s}, \mathbf{S2}, \mathrm{purchasereq}(\lambda^k_{\mathcal{B}_\mathrm{req}})\right\rangle \rightarrow \langle P_\mathrm{s}, \mathbf{S3}, .\rangle$
TR13: $\langle P_\mathrm{s}, \mathbf{S3}, S_\mathrm{p}\rangle \rightarrow \langle P_\mathrm{s}, \mathbf{S4}, .\rangle$
TR14: $\left\langle P_\mathrm{s}, \mathbf{S4}, \rho_{\mathcal{B}_\mathrm{req}}\right\rangle \underrightarrow{\mathbf{L5}} \langle P_\mathrm{b}, \mathbf{B2}, .\rangle$
TR15: $\langle P_\mathrm{b}, \mathbf{B4}, \mathrm{generatepurchasereq}(p_j)\rangle \rightarrow \langle P_\mathrm{b}, \mathbf{B2}, .\rangle$
TR16: $\langle P_\mathrm{b}, \mathbf{B4}, \mathrm{acceptoffer}(p_j)\rangle \underrightarrow{\mathbf{L9}} \langle P_\mathrm{s}, \mathbf{S5}, .\rangle$
TR17: $\langle P_\mathrm{b}, \mathbf{B4}, \mathrm{rejectoffer}(p_j)\rangle \underrightarrow{\mathbf{L7}} \langle P_\mathrm{s}, \mathbf{S2}, .\rangle$
TR18: $\langle P_\mathrm{s}, \mathbf{S5}, \mathrm{accept}(p_j)\rangle \underrightarrow{\mathbf{L10}} \langle P_\mathrm{b}, \mathbf{B5}, .\rangle$
TR19: $\langle P_\mathrm{s}, \mathbf{S5}, \mathrm{reject}(p_j)\rangle \underrightarrow{\mathbf{L8}} \langle P_\mathrm{b}, \mathbf{B2}, .\rangle$

The dialogue game framework and the semantic mechanisms we have presented generate dialogues automatically.[12]

6 A Purchase Scenario

A buyer desires to buy a product of category θ where: $X_\theta = \{price, brand\}$ and $D_\theta = \{[0, 30000], \{A, ..., J\}\}$. The requirements on the attributes are defined in Table 1. The overall satisfaction degree is computed by means of the *min* operator. In the inexpressive profile $N_\mathrm{b} = \{\xi = 0, \eta = \phi\}$, so the buyer agent

Table 1. Requirements on Price and Brand

Price	$\mu_{R_1^\mathrm{f}}$	Brand	$\mu_{R_2^\mathrm{f}}$	Price	$\mu_{R_1^\mathrm{f}}$	Brand	$\mu_{R_2^\mathrm{f}}$
¡6000	1	J	1	[16000, 18000)	0.4	D	0.4
[6000, 8000)	0.9	I	0.9	[18000, 20000)	0.3	C	0.3
[8000, 10000)	0.8	H	0.8	[20000, 22000)	0.2	B	0.2
[10000, 12000)	0.7	G	0.7	[22000, 24000]	0.1	A	0.1
[12000, 14000)	0.6	F	0.6	¿24000	0		
[14000, 16000)	0.5	E	0.5				

will use the *desire_to_buy()* locution to express the purchase requirements. In the expressive profile $N_\mathrm{b} = \{\xi = 1, \eta = 1\}$. In words, the *prefer_to_buy()* locution will be used to express the purchase requirements, and the buyer will attend

[12] To prove this, we need to demonstrate that: every locution can be invoked by one or more semantic mechanisms, and every execution of each of these mechanisms ultimately invokes a locution. We refer to [9] for a detailed example of demonstration.

Table 2. Catalogue of Products

i	Product	Profit	i	Product	Profit
1	{11000, A}	1	7	{21000, I}	0.1
2	{13000, C}	0.1	8	{25000, J}	0.5
3	{15000, D}	1	9	{25000, G}	0.5
4	{17000, E}	0.1	10	{23000, E}	0.5
5	{19000, G}	1	11	{21000, C}	0.5
6	{19000, H}	0.1	12	{19000, A}	0.5

the relax requirements as much as possible. The seller agent owns a catalogue of products which is defined in Table 2. If the seller agent decides to negotiate with an inexpressive profile, $N_{\mathrm{s}} = \{\psi = 0, \beta = \phi, \Delta^k = \phi\}$, so the seller agent does not use explicit relax requirements, and the *refuse_to_sell()* locution is used to request the relaxation of purchase requirements. In the expressive profile, $N_{\mathrm{s}} = \{\psi = 1, \beta = 0.5, \Delta^k = (\delta_1^k = (30000, 1), \delta_2^k = (null, null)), (\gamma_1^k = 1, \gamma_2^k = 0))\}$, so the seller agent uses the *prefer_to_sell()* locution to request the relaxation of purchase requirements. While it has a high degree of certainty regarding the beliefs about the price (reservation value of 30000 and linear estimate of distance), it knows nothing about the brand attribute. These parameters are assumed not to change over time. Table 3 shows the ending of the purchase negotiation dialogues for the expressive and inexpressive profiles. The progress of the negotiation in the first steps is similar because all the potential sale offers are far from the buyer requirements. However, when $\lambda_{\mathcal{B}_{\mathrm{req}}}$ is near to $(17000, E)$ and $(15000, D)$, the seller agent indicates that the brand constraint should be relaxed (stages *S11-12*).[13] The aggregated utility of the outcome is 1.4, while in the inexpressive profiles is 0.5.

In order to test the efficiency of the proposed model, we have developed the following tests which compare the expressive and inexpressive approaches. We assume a static scenario where both the buyer and seller agents are risk averse, and so they try to reach an agreement as soon as possible. This assumption is necessary in order to justify the use of the expressive strategy by the buyer agent.

Each product in the seller's catalogue has 10 attributes, while the FCSP of the buyer agent is defined as a set of 10 fuzzy constraints, where a fuzzy constraint is assigned to each attribute. Each fuzzy constraint has 11 cut levels, all of them equally spaced. In the expressive profile the buyer agent defines $N_{\mathrm{b}} = \{\xi = 1, \eta = 1\}$, and the seller agent $N_{\mathrm{s}} = \{\psi = 1, \beta = 0.5, \Delta^k = (\delta_i^k = (a_i^{\mathrm{res}}, 1)|i = 1...10, \gamma_i^k = 1|i = 1...10)\}$. The set of beliefs state that the certainty degree about all the attribute estimates is 1. The distance estimates are modelled as linear functions, while the reservation value estimates are fixed around the boundaries of the crisp constraints induced at the lowest cut levels. The different experiments are designed varying the seller's catalogue. Each catalogue is an aggregation of

[13] When using the inexpressive negotiation profiles the buyer agent submits *desire_to_buy([¡=18000] & {E-J})*, so the outcome is *(17000,E)* instead of *(15000,D)*.

Table 3. Purchase Negotiation Dialogues

Expressive		**Inexpressive**
prefer_to_buy($\lambda_{\mathcal{B}_{\text{req}}}$) ($\upsilon_{\mathcal{B}_{\text{req}}}$) ⇒	**Bx**	desire_to_buy($\lambda_{\mathcal{B}_{\text{req}}}$)
prefer_to_sell($\lambda_{\mathcal{B}_{\text{req}}}$) ($\rho_{\mathcal{B}_{\text{req}}}$) ⇒	**Sx**	refuse_to_sell($\lambda_{\mathcal{B}_{\text{req}}}$)
...	**...**	...
prefer_to_buy([<=12000]) & {H-J}) ([0.5,0.5])⇒	**B8**	desire_to_buy([<=12000]) & {H-J})⇒
prefer(p_i)={0 0.18039 0.61665 0.14906 0.57807		
0.12807 0.10422 0.25 0.25 0.27805 0.30422 0.32807}		
prefer_ to_sell([<=12000] & {G-J}) ([1,1])⇒	**S8**	refuse_ to_sell([<=12000] & {G-J})⇒
prefer_to_buy([<=14000]) & {G-J}) ([0.55556,0.44444])⇒	**B9**	desire_to_buy([<=14000]) & {G-J})⇒
prefer(p_i)={0 0.19267 0.63563 0.16727 0.5937		
0.1437 0.11572 0.25 0.25 0.28422 0.31572 0.3437}		
prefer_ to_sell([<=14000] & {G-J}) ([1,1])⇒	**S9**	refuse_ to_sell([<=14000] & {G-J})⇒
prefer_to_buy([<=14000]) & {F-J}) ([0.5,0.5])⇒	**B10**	desire_to_buy([<=14000]) & {F-J})⇒
prefer(p_i)={0 0.18572 0.62931 0.16222 0.58991		
0.13991 0.11319 0.25 0.25 0.28296 0.31319 0.33991}		
prefer_ to_sell([<=14000] & {F-J}) ([1,1])⇒	**S10**	refuse_ to_sell([<=14000] & {F-J})⇒
prefer_to_buy([<=16000]) & {F-J}) ([0.54545,0.45455])⇒	**B11**	desire_to_buy([<=16000]) & {F-J})⇒
prefer(p_i)={0 0.1914 0.6414 0.18274 0.60912		
0.15912 0.12808 0.25 0.25 0.2912 0.32808 0.35912}		
prefer_ to_sell([<=16000] & {F-J}) ([0,1])⇒	**S11**	refuse_ to_sell([<=16000] & {F-J})⇒
prefer_to_buy([<=16000]) & {E-J}) ([0.5,0.5])⇒	**B12**	desire_to_buy([<=16000]) & {E-J}) ⇒
prefer(p_i)={0 0.18572 0.63572 0.17769 0.60533		
0.15533 0.12555 0.25 0.25 0.28994 0.32555 0.35533}		
prefer_ to_sell([<=16000] & {E-J}) ([0,1])⇒	**S12**	refuse_ to_sell([<=16000] & {E-J})⇒
prefer_to_buy([<=16000]) & {D-J}) ([0.46154,0.53846])⇒	**B13**	desire_to_buy([<=18000]) & {E-J})⇒
willing_to_sell(15000,D)⇒	**S13**	willing_to_sell(17000,E)⇒
agree_to_buy(15000,D)⇒	**B14**	agree_to_buy(17000,E)⇒
agree_to_sell(15000,D)⇒	**S14**	agree_to_sell(17000,E)⇒

a *feasible set* and a *noise set*. The *feasible set* is the set of products which can be the outcome of a purchase negotiation, while the rest of the catalogue is the *noise set*. If we assume a *min* operator for the calculus of the overall satisfaction degree, the *feasible set* will be formed by the set of products with the highest utility for the buyer agent.[14] We have developed two sets of experiments, with *empty* and *non-empty noise sets*. In both cases we have defined *feasible sets* with different lengths: 2, 5, 10, 25 and 50 products. Given a *feasible set*, next step is to assign a profit u_i to each product in the set. These values have been generated by means of a uniform distribution with parameters 0 and 0.89. Finally we take one of these products and change its profit to 0.9, so this would be the preferred outcome. The *noise sets* are randomly generated and restricted to products with a lower utility for the buyer agent. Table 4 shows the results of the tests. Each

[14] Note that the buyer agent minimizes the loss of potential overall satisfaction degree when it relaxes a purchase requirement, and the seller agent never delay a sale offer waiting for a better agreement.

Table 4. Summary results for the inexpressive and expressive profiles

Inexpressive				
Feasible set length	Success (%)	CI (%)	Median	CI
2	69	[58.97, 77.87]	0.9	[0.8731, 0.9]
5	26	[17.74, 35.73]	0.5970	[0.491, 0.703]
10	15	[8.65, 23.53]	0.5497	[0.4644, 0.6351]
25	3	[0.62, 8.52]	0.5515	[0.499, 0.6039]
50	8	[3.52, 15.16]	0.5418	[0.4702, 0.6133]
Expressive without Noise				
Feasible set length	Success (%)	CI (%)	Median	CI
2	98	[92.96, 99.76]	0.9	[0.9, 0.9]
5	77	[67.51, 84.83]	0.9	[0.9, 0.9]
10	55	[44.73, 64.97]	0.9	[0.8347 ,0.9]
25	28	[19.48, 37.87]	0.7769	[0.7072, 0.8467]
50	24	[16.02, 33.57]	0.7792	[0.7022, 0.8561]
Expressive with Noise				
Feasible+Noise set lengths	Success (%)	CI (%)	Median	CI
2+100	91	[83.60, 95.80]	0.9	[0.9, 0.9]
5+100	46	[35.98, 56.26]	0.8635	[0.7955, 0.9]
10+100	37	[27.56, 47.24]	0.7484	[0.6735, 0.8232]
25+100	35	[25.73, 45.18]	0.6902	[0.6066, 0.7738]
50+100	9	[4.2, 16.4]	0.5819	[0.5144, 0.6494]

test is defined by the use of the expressive or the inexpressive profiles and the length of the feasible and noise sets. For each combination of parameters (profile and lengths), our test system randomly generates 100 different catalogues in order to carry out 100 different negotiations. Then we compute the *success rate* and the *median* of the seller's profit, together with their confidence intervals for a confidence level of 95%. We mean with *success* that the seller agent obtains a profit of 0.9. We can see how the use of expressive profiles significatively improves the results.

7 Conclusions and Future Work

This paper has presented a model to support automated bilateral purchase negotiations using fuzzy constraints. All the internal decision mechanisms, and the rules which govern interactions have been defined. In contrast to other fuzzy constraint models, in ours: (1) A buyer agent attends the seller's requirements in order to select the alternative from the set of trade-off proposals that is likely to benefit both agents. (2) Constraints can be valued in order to help the seller agent to make a more effective search. The purpose of this search is to select the most convenient potential sale offers in order to generate a balanced relax requirement. (3) Different attitudes for the seller and buyer agents can be easily modelled by means of the negotiation profiles.

While this approach has yielded some promising results, considerable work remains to be done. We should identify which are the best strategies to apply under different purchase scenarios. Moreover, the decision mechanisms must be extended in order to cover fuzzy constraints which depend on multiple variables. The future work will also include a deep analysis of the privacy issues, computational efficiency and the robustness of our model.

References

1. Lomuscio, A., Wooldridge, M., Jennings, N.R.: A classification scheme for negotiation in electronic commerce. In: Agent Mediated Electronic Commerce, The European AgentLink Perspective., London, UK, Springer-Verlag (2001) 19–33
2. Luo, X., Jennings, N.R., Shadbolt, N., Leung, H.F., Lee, J.H.: A fuzzy constraint based model for bilateral, multi-issue negotiations in semi-competitive environments. Artificial Intelligence **148**(1-2) (2003) 53–102
3. Lai, K.R., Lin, M.W.: Modeling agent negotiation via fuzzy constraints in e-business. Computational Intelligence **20**(4) (2004) 624–642
4. Kowalczyk, R., Bui, V.: On constraint-based reasoning in e-negotiation agents. In Dignum, F., Corts, U., eds.: Agent-Mediated E-Commerce III. Volume 2003 of Lecture Notes in Artificial Intelligence., Springer (2000) 31–46
5. Castro-Schez, J.J., Jennings, N.R., Luo, X., Shadbolt, N.R.: Acquiring domain knowledge for negotiating agents: a case of study. International Journal of Human-Computer Studies **61**(1) (2003)
6. Luo, X., Lee, J.H., Leung, H.F., Jennings, N.R.: Prioritised fuzzy constraint satisfaction problems: axioms, instantiation and validation. Fuzzy Sets and Systems **136**(2) (2003) 151–188
7. Faratin, P., Sierra, C., Jennings, N.R.: Using similarity criteria to make issue trade-offs in automated negotiations. Artificial Intelligence **142**(2) (2002) 205–237
8. Jennings, N.R., Parsons, S., Noriega, P., Sierra, C.: On argumentation-based negotiation. In: Proc. Int. Workshop on Multi-Agent Systems (1998) 1–7
9. McBurney, P., Euk, R.M.V., Parsons, S., Amgoud, L.: A dialogue game protocol for agent purchase negotiations. Autonomous Agents and Multi-Agent Systems **7**(3) (2003) 235–273
10. McBurney, P., Parsons, S., Wooldridge, M.: Desiderata for agent argumentation protocols. In: AAMAS '02 Proceedings, New York, USA, ACM Press (2002) 402–409
11. Dubois, D., Fargier, H., Prade, H.: Propagation and satisfaction of flexible constraints. Fuzzy Sets, Neural Networks and Soft Computing (1994) 166–187
12. Dubois, D., Fargier, H., Prade, H.: Possibility theory in constraint satisfaction problems: Handling priority, preference and uncertainty. Applied Intelligence **6**(4) (1996) 287–309

Author Index