

The 2011 Workshop on Trading Agents and Design Analysis

TADA-11

<http://issel.ee.auth.gr/tada2011/>

Held in conjunction with the 22nd Int. Conference on
Artificial Intelligence (IJCAI 2011)

17 July 2011

Barcelona - Spain

Message from the Workshop Chairs

We are pleased to welcome you to attend the 2011 Workshop on Trading Agents and Design Analysis (TADA-11), joint with IJCAI2011. We hope you find work published in these proceedings interesting and intriguing.

The study and design of trading agents are prominent application areas in artificial intelligence, since they challenge models of rational decision-making, and have interesting potential benefits for electronic commerce. To date, a broad range of trading scenarios and agent approaches have been studied, creating an extensive and rich research area. This workshop focuses on the design and evaluation of trading agents. Papers on trading agent architectures, decision making algorithms, theoretical analysis, empirical evaluations of agent strategies in negotiation scenarios, game-theoretic analyses, trading mechanisms and market architectures are all within the scope of the workshop.

The TADA workshop series traditionally (from 2003 on) coincides with the finals of the Trading Agent Competition (<http://www.sics.se/tac/>), and many of the contestants participate in the workshop venue. TADA has already attracted more than 30 registrations from the trading agent and auction community.

TADA has a long and successful history being co-located at different artificial intelligence conferences, with IJCAI being the main host. Its co-location with IJCAI is a great opportunity for even greater exposure, since the IJCAI audience will most probably find the new Trading Agent Competition game (TAC Energy – <http://www.tacenergy.org/>) to be of great interest.

The TADA-11 workshop encourages and promotes theoretical and applied research and development, which focuses on:

- Distributed (scalable) algorithmic mechanism design
- Mechanisms for unreliable, dynamic and asynchronous environments
- Mechanisms for incomplete and/or imperfect information environments
- Mechanisms for information goods and services
- Distributed (agent and mechanism) learning models
- Economic and game theoretic specification, design and analysis
- Bargaining, voting and auction mechanisms
- Distributed reputation and trust mechanisms
- Agents that support bidding and negotiation
- Empirical evaluation of human-agent trading
- Simulation and evaluation of properties of novel and complex mechanisms
- Implemented agent-mediated electronic-commerce systems

This year TADA accepted 9 research papers, due to time and space limitations. Given that TADA coincides with the first day of the TAC finals, focus is also given on the TAC community. Nevertheless, work included covers all major TADA axes. Each paper was reviewed by at least three members of the Program Committee, thus ensuring paper quality.

Finally, we appreciate the contributions made by all authors and the effort put by program committee members, workshop and local organizers in order to make TADA'11 a success.

June 2011

Andreas Symeonidis (TADA-11 Workshop Chair)

John Collins (General TAC-11 Chair)

Wolf Ketter (Chair of the Board of TAC organization)

Organization

Workshop Chair

- Andreas Symeonidis Aristotle University of Thessaloniki

General TAC-11 Chair

- John Collins University of Minnesota

Chair of the Board of TAC organization

- Wolf Ketter Rotterdam School of Management

Program Committee

- Michael Benisch Carnegie Mellon University
- Ken Brown University College Cork
- Maria Fasli Essex University
- Shaheen Fatima University of Loughborough
- Enrico Gerding University of Southampton
- Maria Gini University of Minnesota
- Amy Greenwald Brown University
- Sverker Janson Swedish Institute of Computer Science
- Patrick Jordan University of Michigan
- Kate Larson University of Waterloo
- Kevin Leyton-Brown University of British Columbia
- Peter McBurney University of Liverpool
- Pericles A. Mitkas Aristotle University of Thessaloniki
- Tracy Mullen Penn State University
- Jinzhong Niu Brooklyn College, City University of New York
- Benno Overeinder Vrije Universiteit Amsterdam
- David Pardoe University of Texas at Austin
- Steve Phelps University of Essex
- Juan Antonio Rodriguez Aguilar IIIA-CSIC
- Alex Rogers University of Southampton
- Norman Sadeh Carnegie Mellon University
- Alberto Sardinha Carnegie Mellon University
- Peter Stone University of Texas at Austin
- Ioannis A. Vetsikas University of Southampton
- William Walsh CombineNet
- Michael Wellman University of Michigan
- Dongmo Zhang University of Western Sydney
- Haizheng Zhang Penn State University

Table of Contents

Pricing mechanism for real-time balancing in regional electricity markets <i>Mathijs de Weerd, Wolfgang Ketter and John Collins</i>	6
Strategic Formation of Credit Networks: Preliminary Report <i>Pranav Dandekar, Bryce Wiedenbeck, Ashish Goel and Michael P. Wellman</i>	13
Dealing with Trust and Reputation in unreliable Multi-agent Trading Environments <i>Iraklis Tsekourakis and Andreas L. Symeonidis</i>	21
Improving Prediction in TAC SCM by Integrating Multivariate and Temporal Aspects via PLS Regression <i>William Groves and Maria Gini</i>	29
Testing Adaptive Expectations Models of a Continuous Double Auction Market Against Empirical Facts <i>Neil Rayner, Steve Phelps and Nick Constantinou</i>	37
Self-Confirming Price Prediction for Bidding in Simultaneous Second-Price Sealed-Bid Auctions <i>Dong Young Yoon and Michael P. Wellman</i>	44
Rank and Impression Estimation in a Stylized Model of Ad Auctions <i>Jordan Berg, Carleton Corin, Amy Greenwald, and Eric Sodomka</i>	52
An Adaptive Proportional Value-per-Click Agent for Bidding in Ad Auctions <i>Kyriakos C. Chatzidimitriou, Lampros C. Stavrogiannis, Andreas L. Symeonidis and Pericles A. Mitkas</i>	62
A Bidding Agent for Advertisement Auctions: An Overview of the CrocodileAgent 2010 <i>Irena Siranovic, Tomislav Cavka, Ana Petric and Vedran Podobnik</i>	69

Pricing mechanism for real-time balancing in regional electricity markets

Mathijs de Weerd
M.M.deWeerd@tudelft.nl
Delft University of Technology

Wolfgang Ketter
WKetter@rsm.nl
Erasmus University

John Collins
jcollins@cs.umn.edu
University of Minnesota

Abstract

We consider the problem of designing a pricing mechanism for precisely controlling the real-time balance in electricity markets, where retail brokers aggregate the supply and demand of a number of individual customers, and must purchase or sell power at the wholesale level such that the total supply matches total demand. This is typically done for future time periods by buying and selling power, and by setting variable prices for retail customers. In real time, balancing must be done through purchase of regulating services, and by remotely controlling portions of retail customer loads and sources. We enumerate the desirable properties of a market-based balancing mechanism, and analyze the applicability of known theory in two scenarios: a baseline scenario in which brokers have no ability to manipulate their customers' supply and demand, and a single-period scenario with controllable loads. The latter provides promising results for a scenario that takes interactions among time periods into account.

1 Introduction

In electricity markets, supply and demand have to be balanced perfectly in real time. A major task of the Independent Systems Operator (ISO)¹ on the wholesale (transmission) level and of the Distribution Utility (DU) on the regional (distribution) level is to monitor the grid and to maintain balance while keeping voltage, frequency, and power factor within very tight bounds. This task gets more challenging as more renewable energy sources, such as solar and wind, are connected to the grid [Vandezande *et al.*, 2010]. Many of these sources (e.g. wind) are only partially predictable. The grid balancing problem has been studied on various levels (wholesale vs. retail) and with different approaches [Möller *et al.*, 2011].

We study the problem of how to determine the price that energy brokers (retailers) need to pay in case their profile of consumption and production of electricity is imbalanced in

the current time slot.² In a retail electricity market, brokers have the possibility to control some portion of customer production and consumption, such as by offering price concessions in exchange for the ability to remotely interrupt loads or sources for limited periods of time. Examples of such controllable loads are remotely controlled CHPs (Combined Heat and Power – gas turbines that supply both heat and power, and allow some control of production) and domestic water heaters that can be remotely turned off to temporarily reduce consumption.

We present two different scenarios and the related market mechanisms to balance supply and demand: I) without controllable loads, and II) with controllable loads for the current time slot. Our main mechanisms for a regional real-time balancing market are efficient, and create an incentive to resolve imbalances in the day-ahead market. An advantage of our methods is that they are compatible with existing energy market structures (keeping the day-ahead market as a given), and therefore the required restructuring would be less disruptive than with other methods currently suggested.

The paper is organized as follows. In Section 2 we review relevant literature. Section 3 describes our testbed, the Power Trading Agent Competition (Power TAC). In Section 4 we study the problem of setting prices for imbalanced portfolios from the standpoint of mechanism design. Sections 5 and 6 describe the analysis of scenario I and II respectively. Finally, we conclude with directions for future research, and especially discuss some ideas for extending our work to a repeated scenario.

2 Related work

Skytte [1999] gives an econometric analysis of the regulating power market on the Nordic power exchange Nord Pool, and Möller *et al.* [2011] give a nice overview of the structure of wholesale balancing markets in Europe and the USA. A great general overview of the hierarchy, structure, and operation of electricity markets is presented in Shahidehpour *et al.* [2002].

In the Norwegian market there is a given price for electricity in the “regulating market”, which is set to be fixed (β), defined by the wholesale side of the real-time balancing market. This price is always higher than the “spot”

¹In Europe the name TSO (Transmission Systems Operator) is used instead of ISO.

²Energy is traded in predefined fixed intervals; usually 15min or 60min long.

price in the day-ahead market (the most recent price for the current time period in the day-ahead market) in case there is over-consumption (under-production), and always lower when there is under-consumption [Fleten and Pettersen, 2005]. Similarly, in some countries a small fixed payment is given to over-producing, and a quite high penalty is charged to over-consuming brokers.

Although, as we see in this paper, such payments (if set sufficiently high) indeed are able to motivate retail brokers to have a balanced portfolio, they do not exploit the possibility of controllable loads. Recent work by Höning *et al.* [2011a] studies a complete redesign of energy markets to include controllable loads as part of bids in the day-ahead market [Höning *et al.*, 2011b].

Additionally, energy storage can be one of the key underpinnings of the Smart Grid. In [Vytelingum *et al.*, 2010], the authors have developed a framework to analyze agent-based micro-storage management for the smart grid. Specifically, they designed a storage strategy (with an adaptive mechanism based on predicted market prices) for consumers and empirically demonstrated that the average storage profile converges towards a Nash equilibrium. This mechanism can help to maintain balance in a regional grid to the extent that such batteries have sufficient capacity to produce or absorb a significant portion of short-term imbalance.

In [Ketter *et al.*, 2009] the author present the concept of *economic regimes*, an approach to identify the current and predict future market conditions, such as over-supply or scarcity. This approach could better inform individual brokers and consequently used to procure or sell energy ahead of time to avoid future imbalances. For instance, brokers can offer special tariffs to customers with controllable loads, which start their washing or laundry machine automatically when there is extra power (over-supply), e.g. wind, available [Block *et al.*, 2010]. In the next section we discuss *Power TAC*, our agent-based testbed.

3 Testbed: Power TAC

Power TAC [Ketter *et al.*, 2010] (see Figure 1) is a competitive simulation of retail electric power markets that models a market-based management structure for local energy grids. It will challenge research teams from around the world to write autonomous agents, or agent-assisted decision support systems for human operators [Varga *et al.*, 1994], that could operate effectively and profitably in direct competition with each other, while continuously balancing supply and demand in a heterogeneous customer portfolio. Teams will also be challenged to exploit the structure of the market, and that structure will be adjusted and fine-tuned after each competition tournament to defeat counterproductive strategic behaviors and to ultimately develop a set of competitively proven market rules. The result will be a body of valuable research data, along with a much higher degree of confidence that such well-tested market mechanisms could be safely introduced into real world smart grid systems. The competition will also produce a variety of thoroughly tested agent-based energy market trading strategies.

The competition is focused on the role of a broker acting as

an “aggregator” of energy supply and demand, represented by a trading agent. According to Von Dollen [2009] “*aggregators combine smaller participants (as providers or customers or curtailment) to enable distributed resources to play in the larger markets.*”. In reality brokers could be energy retailers, municipal utilities, or cooperatives; in some cases large utilities could also take on the role of brokers.

After a setup period, brokers may trade in the day-ahead wholesale market and adjust variable prices for their customers at any time. At the end of each time slot, a broker receives a performance report giving the supply and demand volume for each of its contracts and tariffs. For each future time slot a broker must maintain a forecast of its total expected supply and demand. Given this information, the broker’s main task is to adjust prices, and to trade in the wholesale market, in order to achieve expected balance.

Simulation time is divided into discrete time slots, each representing a fixed block of time in the simulation timeline. To keep the simulation tractable, power consumption and production are assumed to be constant during a time slot. Before a time slot becomes the “current” time slot, trading for energy in that time slot in the wholesale market is closed. Once a time slot becomes the current time slot, the Distribution Utility (DU) is responsible for achieving balance between supply and demand within the distribution grid while maintaining voltage and frequency within tight bounds; this is equivalent to requiring that the total energy surplus in addition to purchased power is equal to the total energy consumption in addition to sold power, across all brokers. We assume this will be accomplished by technical means (some combination of shedding load, increasing production) combined with participation in the wholesale “real-time” market, which trades in real-time “regulation capacity”. The costs for these services must be passed along to the brokers in a way that motivates them to achieve balance in expectation, and to reveal their true capabilities and costs for shedding loads and increasing production within their customer portfolios.

4 Mechanism design for resolving imbalances

The main goal of a real-time balancing mechanism is to have a balanced system, using the services of the wholesale regulating market, local storage or spinning reserves, or controllable loads and sources made available by brokers, such that demand and supply is matched exactly. To arrive at this goal, we study the problem of setting prices for imbalanced portfolios from the standpoint of mechanism design. Therefore, we first discuss desirable properties of such a pricing mechanism, and we analyze what information is private to the brokers. These properties and the relevant private information differ slightly depending on whether brokers have access to controllable loads and sources. We start with the properties that hold for all scenarios.

1. To be able to restore any imbalance, the DU needs to know about expected imbalances from the brokers ahead of time. Therefore, the mechanism should be *incentive compatible* in the equilibrium of the strategies where brokers truthfully inform the DU of *imbalances*.

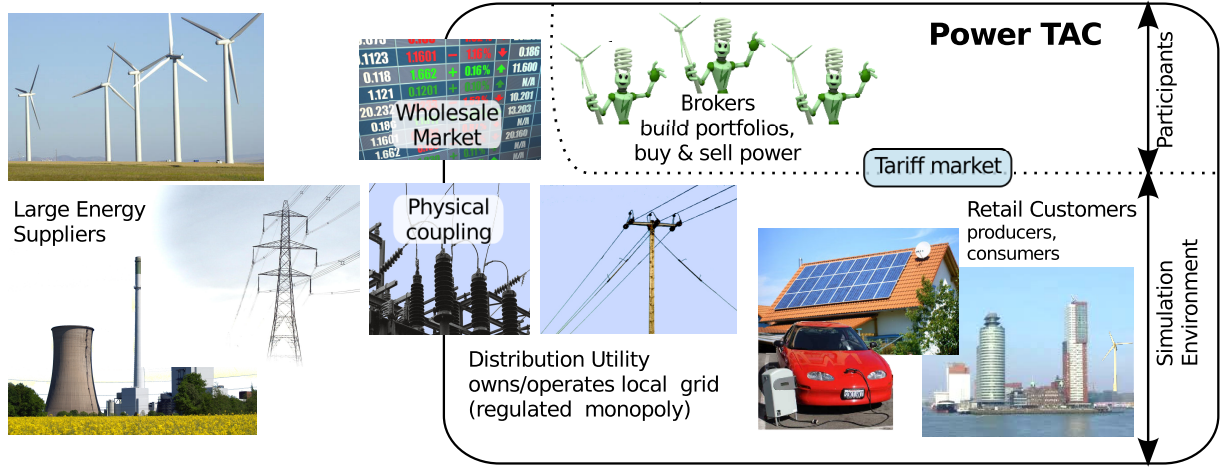


Figure 1: Overview of the Power TAC scenario. In this paper we are primarily interested in the process by which the Distribution Utility achieves real-time balance in the distribution grid.

2. A second desired property is to have an *efficient* system, i.e., which optimizes social welfare. For this we would like to resolve imbalances most cost-effectively, i.e., using the cheapest controllable loads.
3. To arrive at this, we do not just want efficient solutions regarding how imbalances are resolved just in time, we, in fact, would like to have as little imbalance as possible between broker commitments in the day-ahead market and the actual net load experienced in real-time. The idea is that generally more efficient allocations are found when imbalances are resolved in the day-ahead market (or even earlier), simply because there are more options then to produce (or consume) additional power. For example, some generators have a start-up time of several hours. Consequently, the mechanism should be *incentive compatible* in the equilibrium of the strategies where brokers have a portfolio with (almost) *no net imbalance*.
4. Since the DU is responsible for the real-time balancing of the portfolio across all brokers, we can argue that the payments have to be paid to the DU. An additional desired property then is to ensure that the payments offered to the DU are always sufficient to cover its costs. A pricing mechanism meeting this criterion is called *weakly budget balanced*.

In scenario I (without controllable loads) restoring the balance is done solely by the DU. However, to optimally restore the balance when brokers can have controllable loads (scenario II), we need to extract additional information regarding costs and capacities of their controllable loads and sources.

5. Since manipulating the costs of potential controllable loads can lead to sub-optimal solutions, an additional goal in this setting is to design the mechanism such that it is *incentive compatible* in the equilibrium of the strategies of declaring *the true capacities and costs of controllable loads*.
6. A second criterion in the case of controllable loads is

a so-called participation constraint, i.e., the mechanism should benefit participating brokers, or otherwise brokers just will not declare any controllable load at all. In other words, the mechanism should be *individually rational*.

In the following two sections we discuss which mechanisms can be used for the above two scenarios (with/without controllable loads).

5 Scenario I: no controllable loads

The relevant players in both scenarios are the N brokers denoted by $\{1, 2, \dots, n\}$, and the system operator or distribution utility (DU), denoted by 0. In our analysis, as is commonly done, we consider energy production and consumption over a small amount of time, assuming that energy production and consumption are more or less stable during that period. In such a time slot, each broker $i \in N$ has an expected net local energy (potentially negative) surplus of $x_i \in \mathbb{R}$. Each broker is requested to inform the DU of this surplus. Since this *declared surplus* potentially is different from the actual surplus, we denote it by $\hat{x}_i$.³ Furthermore we use P^+ to denote the maximum market price of energy for this time slot in the day-ahead market (over all day-ahead trade periods), P^- to denote the minimum market price over all day-ahead trade periods, and $P^* \geq P$ the all-time highest price possible.

The DU has the potential to import or export energy on the wholesale market (or apply its own ancillary services such as spinning reserves) to arrive at a perfectly balanced energy production and consumption. This comes at a cost of $c_0 : \mathbb{R} \rightarrow \mathbb{R}$ per unit. See Figure 2 for an example, illustrating that the cost of buying additional power is higher than the benefit of selling additional power at this last instance.

³In Power TAC, the declared surplus does not necessarily need to be modeled, since the DU does not actually arrange resources to recover from the net imbalance of all brokers based on the declared surplus.

To make the brokers' strategy of declaring the true imbalance x_i incentive compatible (requirement 1), we apply a mechanism with verification [Nisan and Ronen, 2001]. These mechanisms define payments after the selected outcome (in our case the planned energy production and consumption) has been realized, and there is a possibility of verifying the agents' declaration in this realization; in our case this is the net energy surplus. To obtain incentive compatibility here, we charge a penalty correlated with the difference between the declared and true net energy surplus, which is guaranteed to be higher than the maximum profit that can be made with the same amount of energy in the day-ahead market.

Proposition 1. *A payment of $p_{1,i} = |\hat{x}_i - x_i| \cdot P^*$ ensures that the mechanism is incentive compatible in the equilibrium of declaring the true imbalance (requirement 1).*

Given the declared imbalances, the DU can compute the net declared imbalance $\hat{x} = \sum_{i \in N} \hat{x}_i$ over all brokers, and then apply its (unit) cost function c_0 to the opposite to determine the (expected) total costs for balancing, i.e., $-\hat{x} \cdot c_0(-\hat{x})$. Since in this first scenario there is no other way to recover from imbalances, this meets our requirement (2) of an efficient solution in case all imbalances are declared truthfully.

Proposition 2. *There is only one solution regarding resolving the imbalance, so the mechanism is always efficient (requirement 2).*

In addition to the penalty p_1 set by the mechanism with verification, payments need to be set such that also the third (regarding incentive compatibility) and fourth (regarding budget balancedness) requirement are met. We denote these payments by p_2 . Since these payments are computed after the time slot has passed, we can base them upon the real imbalances x_i . The third requirement in fact implies that the payment for an imbalance should always be higher than resolving the imbalance against the maximum market price P in the day-ahead market, i.e., $p_{2,i} \geq -x_i \cdot P^+$ if $x_i < 0$, or otherwise $p_{2,i} \geq -x_i \cdot P^-$. Finally, the fourth requirement just says the payments from the brokers (who consume more than they produce) should be more than the payments to the brokers (who produce more than they consume) and the costs for recovering from the imbalance together, i.e., $\sum_i p_{2,i} + x \cdot c_0(-x) \geq 0$.

Given these constraints, there are infinitely many possible choices for these payments, since they are only bounded from below. However, we are convinced that a DU should not profit significantly from any imbalances, and the payments should be fair in the sense that brokers that produce too much in an over-consuming market, or brokers that consume too much in an over-producing market should not pay as much as the others. We therefore propose to minimize the difference between the payments and the costs (or profits) attached to resolving the imbalance in the day-ahead market. In the following mathematical programming model, let $p_{2,i}$ denote the payment of broker i ; this is the only variable, since x_i , P^+ ,

and P^- are given.

$$\begin{aligned} & \text{minimize} && \sum_{i \text{ if } x_i < 0} (p_{2,i} + x_i \cdot P^+)^2 + \sum_{i \text{ if } x_i \geq 0} (p_{2,i} + x_i \cdot P^-)^2 \\ & \text{subject to} && p_{2,i} \geq -x_i \cdot P^+ \quad \text{if } x_i < 0 \\ & && p_{2,i} \geq -x_i \cdot P^- \quad \text{if } x_i \geq 0 \\ & && \sum_i p_{2,i} + x \cdot c_0(-x) \geq 0 \end{aligned} \quad (1)$$

This program is a (quadratic) convex program if $c_0(\cdot)$ can be modeled by a (set of) linear function(s); it then can efficiently be solved, e.g., using interior point methods [Boyd and Vandenberghe, 2004].

According to this definition over-consuming ($x_i < 0$) brokers always have to pay a positive amount. In the program in Equation 1 the distribution of the costs of balancing is defined by the minimization criterion, which expresses that each broker should pay an equal portion above the minimum amount defined by the constraints. However, this minimization criterion can be chosen differently (e.g., never let over-producing brokers pay a positive amount); the results below rely solely on the given constraints.

Proposition 3. *An additional payment p_2 as defined in Equation 1 ensures that the resulting mechanism is weakly budget balanced (requirement 4) and incentive compatible in dominant strategies regarding the imbalance in a broker's portfolio (requirement 3).*

Proof. Regarding budget balancedness, $\sum_i p_{2,i} \geq -x \cdot c_0(-x)$ by the last constraint, so the payments received from the brokers exceed the cost for repairing the imbalance. Regarding the incentives, note that the added value of selling an amount $-x_i$ in the day-ahead market at most is $-x_i \cdot P^+$. The net benefit, however, is $-x_i \cdot P^+ - p_{2,i} \leq 0$, since $p_{2,i} \geq -x_i \cdot P^+$. Thus aiming to have $x_i = 0$ dominates having $x_i < 0$. Alternatively, if $x_i > 0$ the payment is never less than $x_i \cdot P^-$, so the net benefit never more than $x_i \cdot P^-$, while selling this amount in the day-ahead would at least have a value of $x_i \cdot P^-$. $\square$

Example 1. *Suppose there are three brokers. Broker 1 has a net surplus of 2 (i.e., $x_1 = 2$), broker 2 has a net surplus of -4, and broker 3 has a perfectly balanced portfolio. In this case, the net surplus is $\sum_i x_i = -2$. Furthermore, assume $c_0(x) = 6$ and the price $P = P^+ = P^-$ in the day-ahead market is 3. The payments according to Equation 1 then are: $p_{2,1} = -4$, $p_{2,2} = 14$, and $p_{2,3} = 2$.*

6 Scenario II: static with controllable loads

The controllable loads for each broker i are represented by a capacity range for its controllable production (or consumption) $[c_i^-, c_i^+]$, and a function describing the price (absolute costs per unit) of diverting from its surplus x_i , i.e., $c_i : [c_i^-, c_i^+] \rightarrow \mathbb{R}$, similar to the up- and downward regulation of the DU. We assume this to be a monotonically increasing (often step-wise) function, since it represents all contracts that include a controllable part, usually at different prices and capacities and first (for rational agents) the cheapest options are used. Examples of such upward regulation contracts are

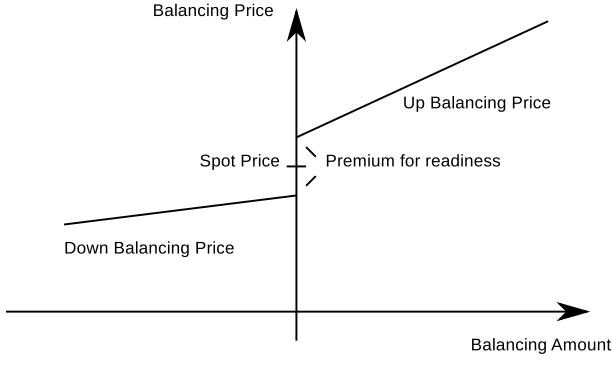


Figure 2: Usually the price (to be received) for a reduction of the production is significantly lower than the price on the spot market, and the price for producing more is significantly higher than the price on the spot market.

the possibility to turn-off lights or heat pumps, or turn on CHPs, and examples of downward regulation contracts are pre-loaded washing machines, the charging of batteries (also of electrical vehicles), and the possibility to temporarily tune down production capacity.

The distribution utility needs to make sure that for every $i \in N \cup \{0\}$ some extra production (or consumption) δ_i within the possibilities is chosen at minimal total costs such that all energy consumption and production is balanced, i.e.,

$$\begin{aligned} & \text{minimize} && \sum_{i \in N \cup \{0\}} \delta_i \cdot c_i(\delta_i) \\ & \text{subject to} && \delta_i \in [c_i^-, c_i^+] \\ & && \sum_{i \in N \cup \{0\}} (x_i + \delta_i) = 0 \end{aligned} \quad (2)$$

This may or may not include an increase/decrease of production regulated by the DU itself, dependent upon the costs. For now we assume this possibility to be unlimited, i.e., $[c_0^-, c_0^+] = [-\infty, \infty]$.

In case the correct information is used, Equation 2 meets the first requirement (of efficiency). Furthermore, if all the functions c_i are monotonically increasing, this problem is convex and can therefore again efficiently be solved [Boyd and Vandenberghe, 2004].

The payments to incentivize the brokers to provide the correct information now consists of three parts:

1. The same mechanism with verification (with payments p_1) as in scenario I is used to make declaring the true imbalance a dominant strategy (see Proposition 1).
2. The same mechanism as in scenario I (with payments p_2) is used to make having no imbalance a dominant strategy.
3. An additional payment p_3 is introduced in this section to make declaring the true cost function a dominant strategy.

The utility of a broker i for a solution δ is not just defined by these payments, but also by the costs of load control, i.e., $u_i(\delta) = -c_i(\delta_i) - p_{1,i} - p_{2,i} - p_{3,i}(\delta)$. Regarding the incentives, note that the first payment is completely independent of the others, and that the analysis of the incentives from the previous section thus automatically transfers. The focus of this section will be on the third payment.

6.1 Real-time matching among brokers

In rare situations (due to irrational/sub-optimal strategies in the day-ahead market) the costs for one broker to produce more energy can be lower than the price another gets for producing less. In that case the above optimization can “match” these to obtain the socially optimal decision. However, for such a scenario (with both buyers and sellers), there is no mechanism that simultaneously satisfies all requirements given above; incentive compatibility (5), individual rationality (6), efficiency (2), and budget balancedness (4) are mutually exclusive in general (see e.g. [Shoham and Leyton-Brown, 2008] for a more extensive overview of the results summarized below.) In theory, for an unrestricted type space, there exists no strongly budget balanced mechanism that implements an efficient choice function in dominant strategies (even in a simple exchange) [Green and Laffont, 1979; Hurwicz and Walker, 1990]. Even if we relax to a Bayes-Nash equilibrium (i.e., the desired strategy is optimal in expectation if all others are rational) and weakly budget balancedness, and if agents are restricted to quasilinear utility functions (i.e., that are linear in the payments), no mechanism is always simultaneously efficient, and (ex interim) individually rational [Myerson and Satterthwaite, 1983].

There are ways around these impossibilities by relaxing some of the requirements. For example, we could use inefficient (but aiming for fairly efficient), (weakly) budget balanced, individually rational, and incentive compatible mechanisms [Barbera and Jackson, 1995; McAfee, 1992], or almost efficient, (weakly) budget balanced, individually rational, and fairly incentive compatible mechanisms [Parkes *et al.*, 2001], or an efficient, (strongly) budget balanced mechanism that is incentive compatible in a Bayes-Nash equilibrium, and only ex ante (i.e., in expectation) individually rational [Arrow, 1979; d’Aspremont and Gérard-Varet, 1979]. However, in the real-time balancing game a very mild assumption allows for an efficient, (ex post, i.e., guaranteed) individually rational mechanism that is incentive compatible in dominant strategies anyway.

6.2 No real-time matching among brokers

First observe that since the verification mechanism can be used to incentivize truthful reporting of the imbalance, the remaining strategic opportunities for brokers relate to the cost function of their controllable loads. However, even then the social costs for balancing can be reduced by real-time matching upward and downward regulating services among brokers. If this is possible, these brokers could also have realized this exchange in the day-ahead market. It turns out that forbidding such exchanges in the real-time balancing phase sufficiently restricts the setting to meet all given requirements. The additional conditions are that for all $i \in N \cup \{0\}$ it holds that

$$\begin{aligned} \delta_i &\geq 0 && \text{if } \sum_{i \in N} x_i < 0 \quad (\text{under-production}) \\ \delta_i &\leq 0 && \text{if } \sum_{i \in N} x_i > 0 \quad (\text{over-production}) \end{aligned} \quad (3)$$

With these conditions and step-wise cost functions, the mechanism is similar to a multi-unit auction (in the case of over-production) or a reverse multi-unit auction (in the case of under-production) [Krishna, 2002]. With any type of cost

i	x_i	$c_i(-1)$	$c_i(1)$
0	-	1	6
1	2	1	5
2	-4	2	4
3	0	3	4

Table 1: An example with the net surplus and costs of controllable production/consumption of three brokers with $c_i^- = -1$ and $c_i^+ = 1$.

functions, Groves [1973] mechanisms are the only mechanisms to achieve both an efficient allocation, in our case, using the least-cost controllable loads, and a truthful declaration, in our case of the cost function of these controllable loads. Within this class, the VCG mechanism [Vickrey, 1961; Clarke, 1971] ensures that brokers always have a nonnegative utility for participating (i.e., individual rationality) under two conditions that hold in this domain: (i) choice-set monotonicity, which says that removing an agent never increases the set of alternative solutions, and (ii) no negative externalities, which says that every agent has zero (or more) utility for any choice that is made without its participation. When VCG is applied in the above setting, given the optimal control vector δ , the payment for each broker i is defined as follows (note that the sign is flipped because VCG is defined on the maximum social welfare, not on the minimal costs).

$$p_{3,i}(\delta) = - \sum_{j \neq i} \delta_j^{-i} \cdot c_j(\delta_j^{-i}) + \sum_{j \neq i} \delta_j \cdot c_j(\delta_j), \quad (4)$$

where δ^{-i} denotes the optimal solution to Equation 2 in a situation where the controllable loads of broker i cannot be used, i.e., $\delta_i = 0$.

Example 2. Continuing upon Example 1, suppose we have the same three brokers. Broker 1 has a net surplus of 2 (i.e., $x_1 = 2$), can regulate 1 unit downward at a profit of $c_1(-1) = 1$, and regulate 1 unit upward at a cost of $c_1(1) = 5$. Similarly, the costs of broker 2 and 3 can be found in Table 1.

The added production δ that optimizes Equation 2 is $\delta = \langle 0, 1, 1 \rangle$, since brokers 2 and 3 are the cheapest. To determine the payments we first determine for each of the brokers what would happen in case the broker's controllable load cannot be used: $\delta^{-1} = \langle \cdot, 1, 1 \rangle$, $\delta^{-2} = \langle 1, \cdot, 1 \rangle$, and $\delta^{-3} = \langle 1, 1, \cdot \rangle$. From this we can then compute the payments regarding the controllable load.

$$\begin{aligned} \pi_{3,1}(\delta) &= -8 + 8 = 0 \\ \pi_{3,2}(\delta) &= -9 + 4 = -5 \\ \pi_{3,3}(\delta) &= -9 + 4 = -5 \end{aligned}$$

In other words, the mechanism pays broker 2 and 3 each 5 for the use of their controllable loads. This completely covers their costs (of 4).

These payments are added to the payments defined in Equation 1 for the imbalances, which are -4, 14, and 2 respectively (see Example 1). Assuming a truthful declaration of the expected imbalances, this comes to a total payment of -4, 9, and -3, respectively. Clearly, $\sum_i p_{2,i} + \sum_i p_{3,i} \geq 0$ in this example.

In case of over-production, all payments are positive, and thus the VCG mechanism meets all our requirements.

Corollary 1. *If there is no real-time matching among brokers, the VCG mechanism is efficient (requirement 2), individually rational (requirement 6), and incentive compatible in dominant strategies regarding the costs of load balancing (requirement 5). In case of over-production the mechanism is weakly budget balanced (requirement 4).*

Proposition 4. *In case of under-production, the mechanism is also weakly budget balanced (requirement 4).*

Proof. Since $c_0(\cdot)$ is taken into account in the mechanism regarding the costs of load balancing, the payments $-p_{3,i}$ never exceed $x \cdot c_0(-x)$, i.e., $-\sum_i p_{3,i} \leq x \cdot c_0(-x)$. From the previous section we know that $x \cdot c_0(-x) \leq \sum_i p_{2,i}$, and thus $\sum_i p_{2,i} + \sum_i p_{3,i} \geq 0$. $\square$

7 Conclusions and future work

We have shown novel applications of known mechanism design theory to regional energy balancing, which is of great importance given the increased desire to integrate renewable energy sources in our existing energy grid. The current paper focuses on the static scenario with controllable loads. We have derived desired properties of a mechanism defining the payments in regional energy balancing. We have showed that declaring the true imbalance can be made a dominant strategy by applying a mechanism with verification, and we have given a bound on the payments related to the price in the day-ahead market to also make having a balanced portfolio a dominant strategy. We have argued that in general the desired properties cannot be met, but we showed how, under mild assumptions in this context, we can apply VCG payments to arrive at an efficient, individually rational, and weakly budget balanced mechanism that is also incentive compatible in dominant strategies.

In practice, however, the process studied is repeated every time slot, and incentive compatibility is not automatically transferred to such a dynamic setting. We leave this even more interesting setting for future work. However, the fact that VCG can be used in the static scenario promises good news for the application of a so-called dynamic-VCG mechanism [Cavallo, 2008].

Additionally, in our further study, we plan to experimentally compare the proposed mechanisms to known mechanisms for setting the payments in the regional balancing phase, such as fixed penalties, or another double auction.

References

- [Arrow, 1979] Kenneth J. Arrow. The property rights doctrine and demand revelation under incomplete information. In M. Boskin, editor, *Economics and Human Welfare*. Academic Press, 1979.
- [Barbera and Jackson, 1995] S. Barbera and M. O. Jackson. Strategy-proof exchange. *Econometrica: Journal of the Econometric Society*, 63(1):51–87, 1995.

- [Block *et al.*, 2010] Carsten Block, John Collins, Sebastian Gottwalt, Wolfgang Ketter, and Christof Weinhardt. Modeling household energy consumption under fixed and variable pricing. In *Workshop on Information Systems and Technology*, St. Louis, Missouri, USA, December 2010.
- [Boyd and Vandenberghe, 2004] S. P. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.
- [Cavallo, 2008] Rugiero Cavallo. *Social Welfare Maximization in Dynamic Strategic Decision Problems*. PhD thesis, Harvard University, 2008.
- [Clarke, 1971] Edward H. Clarke. Multipart pricing of public goods. *Public Choice*, 11(1):17–33, 1971.
- [d’Aspremont and Gérard-Varet, 1979] C. d’Aspremont and L. A. Gérard-Varet. Incentives and incomplete information. *Journal of Public economics*, 11(1):25–45, 1979.
- [Fleten and Pettersen, 2005] Stein-Erik Fleten and Erling Pettersen. Constructing bidding curves for a price-taking retailer in the norwegian electricity market. *IEEE Transactions on Power Systems*, 20(2):701–708, 2005.
- [Green and Laffont, 1979] J. R. Green and J.-J. Laffont. *Incentives in Public Decision Making*. North-Holland, Amsterdam, 1979.
- [Groves, 1973] Theodore Groves. Incentives in teams. *Econometrica*, 41(4):617–631, 1973.
- [Höning *et al.*, 2011a] Nicolas Höning, Han Noot, and Han La Poutré. Integrating power and reserve trade in electricity networks. In *AAMAS ’11: Proceedings of the tenth international joint conference on Autonomous Agents and Multiagent Systems*, 2011.
- [Höning *et al.*, 2011b] Nicolas Höning, Han Noot, and Han La Poutré. On the integration of power and reserve trade in electricity networks. In *ATES ’11: Proceedings of the Agent Technologies for Energy Systems Workshop at AAMAS*, 2011.
- [Hurwicz and Walker, 1990] L. Hurwicz and M. Walker. On the generic nonoptimality of dominant-strategy allocation mechanisms: A general theorem that includes pure exchange economies. *Econometrica*, 58(3):683–704, 1990.
- [Ketter *et al.*, 2009] Wolfgang Ketter, John Collins, Maria Gini, Alok Gupta, and Paul Schrater. Detecting and forecasting economic regimes in multi-agent automated exchanges. *Decision Support Systems*, 47(4):307–318, 2009.
- [Ketter *et al.*, 2010] Wolfgang Ketter, John Collins, and Carsten Block. Smart grid economics: Policy guidance through competitive simulation. Technical Report ERS-2010-043-LIS, RSM Erasmus University, Rotterdam, The Netherlands, 2010.
- [Krishna, 2002] Vijay Krishna. *Auction Theory*. Academic Press, 2002.
- [McAfee, 1992] R. P. McAfee. A dominant strategy double auction. *Journal of Economic Theory*, 56(2):434–450, 1992.
- [Möller *et al.*, 2011] Christoph Möller, Svetlozar Rachev, and Frank Fabozzi. Balancing energy strategies in electricity portfolio management. *Energy Economics*, 33(1):2–11, 2011.
- [Myerson and Satterthwaite, 1983] Roger B. Myerson and Mark A. Satterthwaite. Efficient mechanisms for bilateral trading. *Journal of Economic Theory*, 29(2):265–281, 1983.
- [Nisan and Ronen, 2001] Noam Nisan and Amir Ronen. Algorithmic mechanism design. *Games and Economic Behavior*, 35(1-2):166–196, 2001.
- [Parkes *et al.*, 2001] D. C. Parkes, J. Kalagnanam, and M. Eso. Achieving budget-balance with vickrey-based payment schemes in exchanges. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, pages 1161–1168, 2001.
- [Shahidehpour *et al.*, 2002] Mohammad Shahidehpour, Hatim Yamin, and Zuyi Li. *Market operations in electric power systems: forecasting, scheduling, and risk management*. Wiley-IEEE Press, 2002.
- [Shoham and Leyton-Brown, 2008] Yoav Shoham and Kevin Leyton-Brown. *Multiagent Systems: Algorithmic, Game-theoretic, and Logical Foundations*. Cambridge University Press, 2008.
- [Skytte, 1999] Klaus Skytte. The regulating power market on the Nordic power exchange Nord Pool: An econometric analysis. *Energy Economics*, 21(4):295–308, 1999.
- [Vandezande *et al.*, 2010] Leen Vandezande, Leonardo Meeus, Ronnie Belmans, Marcelo Saguan, and Jean-Michel Glachant. Well-functioning balancing markets: A prerequisite for wind power integration. *Energy Policy*, 38(7):3146–3154, 2010.
- [Varga *et al.*, 1994] László Z. Varga, Nicholas R. Jennings, and David Cockburn. Integrating intelligent systems into a cooperating community for electricity distribution management. *International Journal of Expert Systems with Applications*, 7(4):563–579, 1994.
- [Vickrey, 1961] William Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *Journal of Finance*, 16(1):8–37, 1961.
- [von Dollen, 2009] Don von Dollen. Report to NIST on the smart grid interoperability standards roadmap. Technical Report SB1341-09-CN-0031, Electric Power Research Institute (EPRI), June 2009.
- [Vytelingum *et al.*, 2010] Perukrishnen Vytelingum, Thomas D. Voice, Sarvapali D. Ramchurn, Alex Rogers, and Nicholas R. Jennings. Agent-based micro-storage management for the smart grid. In *Proceedings of 9th International Conference on Autonomous Agents and Multi Agent Systems*, pages 10–14, Toronto, 2010.

Strategic Formation of Credit Networks: Preliminary Report*

Pranav Dandekar
Stanford University
ppd@stanford.edu

Bryce Wiedenbeck
University of Michigan
btwied@umich.edu

Ashish Goel
Stanford University
ashishg@stanford.edu

Michael P. Wellman
University of Michigan
wellman@umich.edu

Abstract

Recent models of abstract credit networks capture the dynamics of trust and obligations among agents over a series of transactions. Such networks support desirable characteristics in a distributed credit model, including robustness to intrusion, bounded risk, and the ability to support wide distributions of transaction pairings even with sparse direct credit relationships. We consider the question of how such networks may be formed in the first place, from decisions of autonomous self-interested agents to grant credit to others. We perform a theoretical analysis of a network contribution game in this framework under a simple scenario, where agents in the network transact only with other agents that they directly trust. Under this restriction we show that at Nash equilibrium, agents allocate credit budgets in a socially optimal manner. We also investigate via simulation another version of the network formation game, where agents may transact along paths in the credit network. In a model where agents have common beliefs about the default risk of others, we find that in equilibrium of the empirical game everyone issues credit to the same agents, yielding a network exhibiting a small set of central currencies.

1 Introduction

The study of strategic network formation seeks to understand the emergent behavior and properties of a network when agents establish connections to other agents in order to maximize their utility. Establishing a connection incurs a cost but also yields some value to agents connected through that edge. The agents are deemed to be utility-maximizing, that is, they make decisions in order to maximize the difference between their total value and their total cost. This problem has been studied in many different settings [Jackson and Wolinsky, 1996; Fabrikant *et al.*, 2003; Corbo *et al.*, 2006; Anshelevich and Hoefer, forthcoming]. One can ask interesting questions about the emergent properties of the networks formed in each setting: What topologies are feasible in equilibrium? What topologies

maximize overall social welfare? How does this depend upon the cost of forming an edge and the value derived from having a connection? What is the price of anarchy? Can agents discover stable states quickly through some kind of iterated better-response dynamics?

This paper is a preliminary investigation into some of these questions in the context of *credit networks*. Credit networks represent a way of modeling trust and managing obligations among entities in a network. Nodes in the network print their own currency and maintain credit relationships representing the amount of each other's currency they are committed to accept. These amounts can be viewed as the degree of trust one agent has for another. Credit commitments allow the network to serve as a decentralized payment infrastructure: arbitrary payments can be routed through the network by passing IOUs between trusting nodes in their respective currencies, thus obviating the need for a common currency. This model was introduced independently by DeFigueiredo and Barr [2005], Ghosh *et al.* [2007], and Karlan *et al.* [2009] as a generalization of budget constraints and as a mechanism for enabling transactions among untrusting agents in a network.

Prior work shows that a network with a reasonable amount of connectivity (liquidity) is sufficient to support and sustain transactions generated throughout the network [Dandekar *et al.*, 2011]. In the current work we address how credit networks may be formed in the first place. We endogenize network formation by explicitly modeling the decision by each agent to issue credit to others. Issuing credit entails risk (a counterparty may violate the trust extended), as well as benefits (increases the probability that profitable transactions may be completed). A natural question is whether agents who rationally weigh these risks and benefits will actually form viable credit networks.

In this paper we provide some preliminary results on strategic formation of credit networks, through two approaches. First, we give theoretical results for a special case of the formation scenario, where agents' trust in each other is captured via an underlying graph. An agent trusts its neighbors in the graph and is therefore willing to extend them credit. The agents' tolerance for exposure to risk in this model is captured via per-node *credit budgets*: the maximum credit that a node is willing to extend to others. An agent's problem is to allocate its credit budget across its neighbors in order to maximize the long-term success probability of their transactions. We

*This work was supported in part by the ARL Collaborative Technology Alliance on Network Science.

find that under certain technical conditions, including a restriction of transactions to occur directly between the parties receiving and issuing credit, the Nash equilibrium solutions correspond exactly to socially optimal credit allocations. Second, we investigate through empirical game-theoretic analysis an alternative model that allows transactions along paths of credit relationships. Here the agents' problem is to choose how much credit to issue, and to whom, to balance the benefits of transaction profitability against the risk of credit default captured via per-node default probabilities. In this preliminary analysis we consider only a particular configuration of environment parameters and a set of eight heuristic strategies. We find that with a common prior on default probabilities (i.e., all nodes have identical estimates of the default probability of a given node), there is a strong tendency to form networks where everyone issues credit to the same small set of the most trustworthy nodes in the network. The networks that emerge in this setting are structurally similar to the centralized network described by Dandekar *et al.* [2011].

Both our theoretical and empirical results are preliminary, but they point to potentially important directions for a full investigation. Our theoretical analysis suggests the possibility that there may be a wider set of scenarios in which the *price of anarchy* of a credit network obtained endogenously by rational agents may be small compared to a centrally designed credit network. Our empirical analysis suggests an important direction for further modeling and investigation: if nodes do not have a shared prior on default probabilities, what kinds of credit networks emerge and how do they differ from a centralized currency system?

2 Background: A Model of Credit Networks

A credit network $G = (V, E)$ is a directed graph with n nodes and m edges. Nodes represent entities or agents. Edges represent pairwise credit limits between agents. An edge $(u, v) \in E$ has capacity $c_{uv} > 0$, which implies that u has extended a credit line of c_{uv} units to v in v 's currency. If a node y needs to pay p units in its currency to node z (for example, to buy a good that z is selling), the payment can go through if the maximum credit flow from z to y is at least p units. The payment will get routed through a chain of nodes from y to z , where each link on the chain carries at least the requisite credit capacity. If an edge (u, v) routes p units of payment from u to v , the credit capacity c_{uv} increases by p while c_{vu} decreases by p . This update is illustrated by Figure 1. Thus routing payments in credit networks is identical to routing residual flows in general flow networks. Note that payment flows in the opposite direction of credit, so a payment merely results in a redistribution of credit; the total credit in the network remains unchanged.

This model is useful in settings where there is a need to model trust between nodes in a network. It is particularly well-suited for transactions in exchange economies such as P2P networks where it can be used to improve inefficiencies resulting from asynchronous demand and bilateral trading [Liu *et al.*, 2010]. It has been used to thwart spam in e-mail and content-sharing systems such as YouTube [Mislove *et al.*, 2008]. It can also be used in settings such as packet

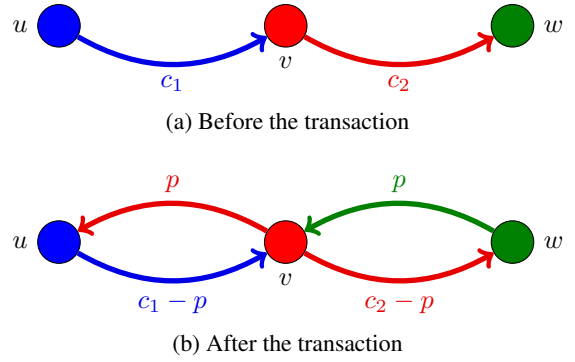


Figure 1: Updating the credit network to process a transaction between u and w worth p units.

routing in mobile ad-hoc networks and combating spam in viral marketing over social networks. There is a large body of work in economics and sociology on social capital and favor exchanges in networks [Jackson *et al.*, 2010]. This model provides a rigorous way of not only keeping track of favors owed to and by each individual in a network, but also a way to exchange favors, via trusted intermediaries, between individuals who do not know each other directly.

A natural question to ask is: How do nodes determine how much credit, if any, to extend each other under some notion of rationality? We assume a model of repeated transactions: nodes in a credit network probabilistically transact with each other at each time step. So the credit extended by a node u to a node v ultimately provides liquidity for node u as well. That is, in addition to enabling v to make payments by giving IOUs to u , it may also enable u to make payments to v since v will take its own IOUs back. On the other hand, when u extends, say, p units of credit to v , u exposes itself to the risk that v might disappear leaving u holding up to p units of v 's irredeemable IOUs. Thus each node is faced with the strategic question of extending credit to maximize liquidity for itself while minimizing the risk it exposes itself to. We model risk in two different ways: in the theoretical analysis (Section 3) we model it as per-node budget constraints (each node has a limit on how much total credit it is willing to extend others), whereas in the simulations (Section 4) we model it as per-node default probabilities so that nodes try to minimize the expected loss from the default of their neighbors.

3 Theoretical Analysis

In this section we present a theoretical analysis of the credit network formation problem by formulating it as a *network contribution game* [Anshelevich and Hoefer, forthcoming]. We show that if we allow only symmetric bilateral transactions, that is, routing payments only along the direct edge between two transacting nodes, and interpret the long-term transaction success probability between adjacent nodes as their edge utilities, then for a broad class of stochastic transaction regimes the credit network formation problem is a network contribution game with concave, differentiable utilities. As a result, the price of anarchy in this game is 1. This means that, informally speaking, agents making selfish decisions based on local in-

formation attain the same social welfare in equilibrium that a central planner (benevolent dictator) could hope to achieve by coordinating the actions of all agents.

3.1 Network Contribution Game

We frame the credit network formation problem as a network contribution game. Our setup is identical to that of Anshelevich and Hoefer [forthcoming], except we describe it in the language of credit networks. We take $G(V, E)$ as an exogenously defined simple undirected graph representing the underlying network. The n nodes in V represent agents, and the m edges in E represent relationships (e.g., friendships) between agents. Each node is constrained to extend credit only to nodes it trusts—its neighbors in G —in order to maximize its total utility. Extending credit to neighboring nodes transforms the given undirected graph G into the directed graph with associated credit capacities described in Section 2. We model risk in terms of the maximum total credit that a node is willing to extend across its neighbors. Each node u has an exogenously defined *credit budget*, $B_u > 0$, which represents the risk that node u is willing to expose itself to. Nodes with higher budgets are more tolerant of risk, and vice versa.

More formally, node u 's strategy is a set of initial credit allocations $\{c_{uv}^0 \mid (u, v) \in E\}$ such that $c_{uv}^0 \geq 0$ and $\sum_{(u,v) \in E} c_{uv}^0 \leq B_u$. Let $\mathbb{R}_+$ be the set of nonnegative real numbers. Each relationship $e = (u, v)$ contributes utility $f_e : \mathbb{R}_+ \times \mathbb{R}_+ \rightarrow \mathbb{R}_+$ to each node u and v , where $f_e(\cdot, \cdot)$, is a non-decreasing function of c_{uv}^0 and c_{vu}^0 . Note that the utility functions on any two edges may be different. Given the initial credit allocations of all nodes, the total utility of a node u , w_u , is simply the sum of utilities on each of its edges, that is, $w_u = \sum_{e=(u,v) \in E} f_e(c_{uv}^0, c_{vu}^0)$. A node u distributes its credit budget B_u among its neighbors in G in order to maximize w_u . We measure the *social welfare*, w_G , of the network as the sum of utilities of all nodes in the network: $w_G = \sum_{u \in V} w_u$. It is easy to show that if the edge utilities, f_e , are concave and differentiable, this game has a price of anarchy of 1.

Theorem 1. *If the edge utilities, $f_e(\cdot, \cdot)$, $e \in E$, are concave and differentiable, the network contribution game has a price of anarchy equal to 1.*

Proof. We know that network contribution games are potential games [Anshelevich and Hoefer, forthcoming]. In particular, the function $\Phi_G := w_G/2 = \sum_{e=(u,v) \in E} f_e(c_{uv}^0, c_{vu}^0)$ is a potential function. We also know that any Nash equilibrium of a potential game with a smooth concave potential is also a potential maximizer [Neyman, 1997]. If edge utilities, $f_e(\cdot, \cdot)$, are concave and differentiable, Φ_G is concave and differentiable. Therefore, any Nash equilibrium of the game must maximize Φ_G . In other words, the price of anarchy of the game is 1. $\square$

The price of anarchy being 1 implies that Nash equilibria of the network contribution game are also optimal in terms of social welfare.

Note that we have not yet specified how nodes derive utility from extending credit to others. We would like to posit that nodes extend credit to each other in order to maximize the

long-term success probability of their transactions. In order to do that, we first describe the model of repeated transactions under which we analyze the credit formation problem.

3.2 Transaction Model

We assume that each edge $e \in E$ is associated with a probability density function $g_e(\cdot)$ having support over $[0, \infty)$ and a corresponding cumulative distribution function $F_e(\cdot)$. Let $F_e(\cdot)$ be twice differentiable. The values of transactions along e are drawn from the distribution given by $g_e(\cdot)$. Additionally, the transaction rate matrix $\Lambda = \{\lambda_{uv} : u, v \in V\}$ is given and satisfies the following properties:

- For all nodes $u \in V$, $\lambda_{uu} = 0$.
- For all nodes $u, v \in V$, $\lambda_{uv} > 0$ if and only if $(u, v) \in E$.
- For all nodes $u, v \in V$, $\lambda_{uv} = \lambda_{vu}$.
- And finally, $\sum_{u,v} \lambda_{uv} = 1$.

We allow only bilateral transactions, that is, routing payments only along the direct edge between two transacting nodes. At each time step t , we pick a node pair $\langle u, v \rangle$ with probability λ_{uv} . We draw a transaction value, say x^t , from the distribution given by $g_e(\cdot)$ and try to route x^t units of flow from u to v along edge e (the flow is presumed to be a payment from u to v). A transaction succeeds if $x^t \leq c_{vu}^t$ and fails otherwise. As described in Section 2, a successful transaction at t results in $c_{uv}^{t+1} \leftarrow c_{uv}^t + x^t$ and $c_{vu}^{t+1} \leftarrow c_{vu}^t - x^t$. Note that for all $t > 0$, $c_{uv}^t + c_{vu}^t = c_{uv}^0 + c_{vu}^0 = c(e)$. For the rest of the section we denote by λ_e the transaction rate between nodes u and v connected by edge e . These repeated probabilistic transactions along edge e induce a Markov chain over $[0, c(e)]$ which is governed entirely by λ_e and $g_e(\cdot)$.

We will show that the steady-state transaction success probability, q_{uv} , between adjacent nodes u and v is an increasing, differentiable function of the total credit capacity, $c(e)$, along edge $e = (u, v)$. As a result, we can treat transaction success probability as a utility measure,

$$f_e(c_{uv}^0, c_{vu}^0) := q_{uv}(c_{uv}^0 + c_{vu}^0).$$

Further, we will show that that if $g_e(\cdot)$ is non-increasing, $q_{uv}(\cdot)$ is concave. Therefore, the credit formation problem in this setting can be cast as a network contribution game with concave, differentiable utilities.

3.3 Transaction Success Probability

We denote the Markov chain along edge e by $\mathcal{M}_e(\lambda_e, g_e) = \{X_e(t) \mid t \geq 0\}$. A state $X_e(t) = x$ of the Markov chain can be interpreted as $c_{uv}^t = x$, $c_{vu}^t = c(e) - x$. Let P_e be the transition kernel of the Markov chain and let p_e be the corresponding density. Then p_e is given by

$$\forall x, y \in [0, c(e)], p_e(x, y) = \begin{cases} \lambda_e g_e(|x - y|) & \text{if } x \neq y \\ (1 - \lambda_e) + \lambda_e \int_x^\infty g_e(y) dy + \lambda_e \int_{c-x}^\infty g_e(y) dy & \text{if } x = y \end{cases}$$

Note that for all $0 \leq x, y \leq c(e)$, $p_e(x, y) > 0$. It implies that $\mathcal{M}_e$ is irreducible.

Lemma 2. Consider an edge $e \in E$. The Markov chain $\mathcal{M}_e(\lambda_e, g_e)$ is irreducible.

Proof. Observe that for any $x \in [0, c(e)]$ and any closed interval $A \subseteq [0, c(e)]$ of nonzero length, $P_e(x, A) > 0$. $\square$

Lemma 3. Consider an edge $e \in E$. The Markov chain $\mathcal{M}_e(\lambda_e, g_e)$ has a uniform steady-state distribution over $[0, c(e)]$.

Proof. Consider a uniform distribution given by π_e such that $\pi_e(x) = 1/c(e)$ for all $x \in [0, c(e)]$. Then, π_e satisfies the detailed balance condition, $\pi_e(x)p_e(x, y) = \pi_e(y)p_e(y, x)$ for all $0 \leq x, y \leq c(e)$. Therefore, π_e is a steady-state distribution of $\mathcal{M}_e$. Further, since $\mathcal{M}_e$ is irreducible, π_e is the unique steady-state distribution of $\mathcal{M}_e$. $\square$

Let $q_{uv}(c)$ be the steady-state success probability of the transaction where u pays v when the total credit capacity on edge (u, v) is c .

Theorem 4. For adjacent nodes u, v such that $e = (u, v)$,

$$q_{uv}(c) = \frac{\lambda_e}{c} \int_0^c F_e(y) dy.$$

Proof. Fix a time step t . Assume that $X_e(t-1) = y$, where $0 \leq y \leq c$. Then the transaction at time step t where u pays v would succeed if $x^t \leq c - y$. So the success probability at time step t is given by $\lambda_e \mathbb{P}[x^t \leq c - y] = \lambda_e F_e(c - y)$. Therefore, the steady-state success probability is given by

$$\begin{aligned} q_{uv}(c) &= \int_0^c \pi_e(y) \lambda_e F_e(c - y) dy \\ &= \frac{\lambda_e}{c} \int_0^c F_e(c - y) dy, \end{aligned}$$

since $\pi_e(y) = 1/c$ for all $0 \leq y \leq c$. Substituting $x = c - y$, we get

$$q_{uv}(c) = -\frac{\lambda_e}{c} \int_c^0 F_e(x) dx = \frac{\lambda_e}{c} \int_0^c F_e(x) dx.$$

$\square$

Since Λ is symmetric, $q_{uv}(c)$ is equal to $q_{vu}(c)$. Note that since $F_e(\cdot)$ is increasing over $[0, \infty)$, $q_{uv}(\cdot)$ is also increasing.

Theorem 5. For an edge $e = (u, v)$, if $g_e(\cdot)$ is non-increasing, $q_{uv}(\cdot)$ is concave.

Proof. Differentiating $q_{uv}(c)$ with respect to c , we get

$$\frac{dq_{uv}}{dc} = \lambda_e \left(-\frac{1}{c^2} \int_0^c F_e(y) dy + \frac{1}{c} F_e(c) \right) \quad (1)$$

$$= \frac{\lambda_e}{c} \left(F_e(c) - \frac{1}{c} \int_0^c F_e(y) dy \right). \quad (2)$$

Note that

$$\begin{aligned} \int_0^c (c - y) g_e(y) dy &= c \int_0^c g_e(y) dy - \int_0^c y g_e(y) dy \\ &= c F_e(c) - \int_0^c y g_e(y) dy \\ &= c F_e(c) - \left(y F_e(y) \Big|_0^c - \int_0^c F_e(y) dy \right) \\ &= \int_0^c F_e(y) dy. \end{aligned}$$

Substituting $\int_0^c (c - y) g_e(y) dy$ for $\int_0^c F_e(y) dy$ in (2), we get

$$\begin{aligned} \frac{dq_{uv}}{dc} &= \frac{\lambda_e}{c} \left(F_e(c) - \frac{1}{c} \int_0^c (c - y) g_e(y) dy \right) \\ &= \frac{\lambda_e}{c} \left(F_e(c) - \frac{1}{c} \int_0^c c g_e(y) dy + \frac{1}{c} \int_0^c y g_e(y) dy \right) \\ &= \frac{\lambda_e}{c} \left(F_e(c) - F_e(c) + \frac{1}{c} \int_0^c y g_e(y) dy \right) \\ &= \frac{\lambda_e}{c^2} \int_0^c y g_e(y) dy. \end{aligned}$$

Differentiating again, we get

$$\begin{aligned} \frac{d^2 q_{uv}}{dc^2} &= \lambda_e \left(\frac{-2}{c^3} \int_0^c y g_e(y) dy + \frac{1}{c} g_e(c) \right) \\ &= \lambda_e \left(\frac{-2}{c^3} \int_0^c y g_e(y) dy + \frac{2}{c^3} g_e(c) \int_0^c y dy \right) \\ &= \frac{2\lambda_e}{c^3} \int_0^c y (g_e(c) - g_e(y)) dy. \end{aligned}$$

So, if $g_e(y) \geq g_e(c)$ for all $y \leq c$, then $\frac{d^2 q_{uv}}{dc^2}$ is non-positive and therefore $q_{uv}(\cdot)$ is concave. $\square$

Note that many natural distributions have non-increasing density functions over $[0, \infty)$. Examples include the exponential distribution, the normal distribution $\mathcal{N}(0, \sigma^2)$, and the power-law distribution.

4 Simulation-Based Game Analysis

In this section we present an experimental framework for evaluating another variant of the credit network formation problem. To explore the issue of trust in credit networks, we consider a scenario where there is explicit risk associated with issuing credit, namely that agents might default, exhausting their credit and leaving the network. Default in this scenario is not a strategic choice, but rather a probabilistic event like network payments. The strategic decision we focus on is that of the agents to issue initial credit. Our model is thus a one-shot game, in which every agent determines its credit allocation, upon which the stochastic sequence of payment and default events transpire according to specified probability distributions.

4.1 Environment

In our credit network game, a set of agents V and a common knowledge event distribution D are exogenously specified. D gives probabilities for two types of interactions among players: credit events and default events. At each time step t , either a credit event or a default event occurs: $e(t) \in \{c, d\}$. A credit event $\langle b^t, s^t, x_{b^t}, x_{s^t} \rangle$ specifies a buyer and seller $b^t, s^t \in V$, as well as a cost x_{s^t} to the seller of providing a good or service and a value x_{b^t} to the buyer of receiving it. If paths through the credit network connecting s to b have total capacity $c^t \geq x_{s^t}$, they transact, transferring credit x_{s^t} . A default event specifies a single agent $d^t \in V$, who exhausts all available credit and leaves the network.

In the experiments described below, seller cost is constant, $x_{s^t} = 1$, for all agents s and all time steps t , and buyer value is drawn uniformly $x_{b^t} \sim U[1.0, 1.2]$. At each time step, one event transpires: a default event with probability $\Pr(e(t) = d) = 0.002$, or a credit event with probability $\Pr(e(t) = c) = 0.998$. In a default event, one agent d^t is selected according to a known probability. This probability comes from assigning each agent a weight drawn uniformly from $[0, 1]$ and normalizing these weights to sum to 1. These probabilities are not updated as agents leave the network, so if an agent that has already defaulted gets selected, no agent defaults in that time step. In a credit event, a pair of agents (b^t, s^t) is selected according to a known probability distribution. This probability comes from assigning each pair a weight drawn from a Pareto distribution with shape parameter $\alpha = 2$.

An action for agent $v \in V$ consists of an initial credit allocation c_{vu}^0 for every other agent $u \in V$. Once credit allocations are determined, agent interactions are simulated for 10,000 time steps according to the event distribution. An agent's payoff after a series of events is the total value of services received minus services provided across all credit events, less any losses from default events,

$$\text{payoff}(v) = \sum_{t|b^t=v} x_{b^t} - \sum_{t|s^t=v} x_{s^t} - \sum_{t|d^t=u} c_{vu}^t.$$

Note that agents are not credited with the amount of the links they exhaust upon defaulting. This payoff varies according to the particular draws from the event distribution, so in our study we simulate the model many times and estimate payoffs by the sample average.

4.2 Empirical Game-Theoretic Analysis Method

Because this model is intractable to solve analytically, we undertake an empirical game-theoretic analysis (EGTA) [Wellman, 2006]. In the EGTA approach, we use simulation to estimate a game form over a set of heuristic strategies. A strategy for the credit network formation game can be any function from the event distribution to an initial credit allocation. To render simulation-based analysis tractable, we enumerate a manageable number of potentially interesting heuristic strategies, as described in Section 4.3.

For an N -player symmetric game over S strategies, there are $\binom{N+S-1}{N}$ distinct profiles. Our simulated credit network has 60 nodes, each corresponding to an agent. Since a 60-player game presents a computationally infeasible profile

space for even a few strategies, we abstract the model to a simpler five-player game using hierarchical reduction [Wellman *et al.*, 2005]. In this reduction, each player of the game chooses a strategy to be implemented by a 12-agent coalition in the original network. Because the players choose heuristic strategies before their types are known, the game is in fact symmetric, and we can find a symmetric mixed-strategy Nash equilibrium using replicator dynamics [Cheng *et al.*, 2004].

To estimate the empirical game, we computed the payoffs for a given profile by drawing an assignment of default weights for each agent and transaction weights for each pair. We then apply the agents' strategies to determine initial credit assignments, and simulate credit and default events according to D for 10,000 time steps on the resulting credit network. The payoff for each profile is estimated by the average over 30 such simulation runs.

4.3 Strategies

We conducted three empirical game experiments. The first included the following five strategies:

- **Zero Credit** The Zero strategy extends no credit, but can still participate in the credit network as long as other agents extend credit to it. Agents following this strategy will never have a negative payoff, but could miss out on many potential transactions.
- **Random Links** The Random strategy extends $c = 5$ units of credit to each other agent with probability $\frac{1}{4}$. This strategy extends significantly more credit links and issues significantly more total credit than any of the others we tested.
- **Low Default Risk** The Low_Default strategy extends 5 units of credit to each other agent whose probability of defaulting is below a threshold (specifically, agents v such that $\Pr(d^t = u | e(t) = d) < 0.004$, meaning that over the course of the 10,000 event simulation run v has less than an 8% chance of ever defaulting). This strategy takes the most conservative step away from Zero, allowing some liquidity with minimal risk.
- **High Transaction Probability** The High_Transact strategy extends 5 units of credit to each other agent with whom the creditor has a high probability of transacting as a buyer ($\Pr(b^t = u, s^t = v | e(t) = c) > 0.0005$, meaning that at least 5 transactions where u buys from v are expected over the 10,000 event simulation run). This strategy seeks to create liquidity to enable the trades the agent considers most probable.
- **High Expected Value** The High_EV strategy extends 5 units of credit to the agents from whom the creditor expects to gain the most value. This value for agent u of issuing credit c to agent v is calculated as the expected value of transactions where u buys from v , minus the expected cost of transactions where u sells to v , minus an estimate of the loss from the possibility that v defaults:

$$\begin{aligned} \mathbb{E}[U_{u,v}] = & \sum_{b^t=u, s^t=v} x_{b^t} - \sum_{s^t=u, b^t=v} x_{s^t} \\ & - c_{uv}^0 \cdot \Pr(v \text{ defaults}), \end{aligned}$$

where $c_{u,v}^0 = 5$ is the initial amount of credit that would be issued. The last term does not perfectly characterize the cost of defaults, because the actual amount of outstanding credit at the time of default can vary based on any transaction in the network which uses this link. In the first experiment, the threshold for issuing credit is set at $\mathbb{E}[U_{u,v}] > 0.5$.

The second experiment added the following two strategies, for a total of seven:

- **High Transact and Low Default** The $\text{HT} \wedge \text{LD}$ strategy starts by eliminating all agents with very high probability of defaulting (the threshold is set much higher than for `Low_Default`, at $\Pr(v \text{ defaults}) < 0.33$ over 10,000 rounds), and then issues 5 units of credit to those among the remaining agents those with whom the creditor is most likely to transact ($\Pr(b^t = u, s^t = v \mid e(t) = c) > 0.0004$, or at least four transactions expected).
- **High EV and Low Default** The $\text{EV} \wedge \text{LD}$ strategy eliminates the same high-default-risk agents from consideration, but then uses the method as `High_EV` to select which of the remaining agents to issue 5 units of credit. This strategy sets its threshold lower than `High_EV`, at $\mathbb{E}[U_{u,v}] > 0$.

The third experiment involved just four strategies: `Zero`, `Random`, `Low_Default`, and $\text{EV} \vee \text{LD}$:

- **High EV or Low Default** The $\text{EV} \vee \text{LD}$ strategy offers 5 units of credit to agents with especially low default probability ($\Pr(d^t = u \mid e(t) = d) < 0.002$, or at most 4% cumulative default probability over 10,000 rounds) and also to agents with especially high expected net transaction value ($\mathbb{E}[U_{u,v}] > 4$).

Note that all agents that issue credit do so in amounts of exactly 5 units, and that the various threshold parameters of those strategies (excluding `Random`) were calibrated so that on average each agent would issue credit to 5–7 others. We intend to loosen these restrictions in future experiments.

4.4 Results

In the first five-strategy experiment, replicator dynamics converged (from any initial mixture) to a pure strategy Nash equilibrium in which all players chose `Low_Default`. When all players adopt this strategy, all agents extend credit to the same small group of highly trustworthy agents (usually 5–7 of them). The resulting network, pictured in Figure 2, bears some resemblance to the centralized currency credit network described by Dandekar *et al.* [2011], in that agents give a large amount of credit to the central subgraph, and transactions can occur only using the currency of one of those central agents.

Following this result, we ran a second experiment which added two strategies that tried to improve on `Low_Default`. These strategies use default probability as a filter to eliminate high-risk agents, and then apply criteria of transaction probability or expected value to make a final selection of how to allocate credit. However, we once again found that replicator dynamics converged to an equilibrium of everyone playing `Low_Default`. This suggests that the value of playing `Low_Default` stems not only from avoiding the cost of

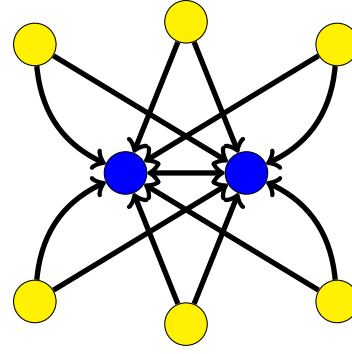


Figure 2: Stylized equilibrium network: all agents follow the `Low_Default` strategy. The result is coordination on a small set of central currencies, represented by the (blue) nodes with incoming arrows.

defaults, but also from how it enables agents to coordinate in connecting to the same subgraph.

In the third experiment, we attempted again to find a strategy that would improve on `Low_Default`. The idea behind the $\text{EV} \vee \text{LD}$ strategy was that it might allow much the same networks stability as `Low_Default`, while potentially increasing liquidity on highly profitable paths. But again we find that only `Low_Default` is played with positive probability in equilibrium.

For all three of these empirical games we measured NE regret [Jordan, 2010], the amount lost by one player when deviating to a given strategy from an ϵ -Nash equilibrium, in this case, the equilibrium in which all players choose `Low_Default`. We also measured the payoff and the regret of each pure strategy profile consisting entirely of one strategy. The regret of a pure strategy profile is the maximum gain a player could achieve by deviating from that profile to any other strategy. The results are presented in Table 1, with strategies ranked by NE regret.

heuristic strategy	NE regret	s.p. regret	s.p. payoff
<code>Low_Default</code>	0.0	0.0	6.8
$\text{HT} \wedge \text{LD}$	0.2	0.9	6.2
$\text{EV} \wedge \text{LD}$	0.8	1.0	6.6
$\text{EV} \vee \text{LD}$	0.9	0.9	6.8
<code>High_Transact</code>	2.0	3.5	4.1
<code>High_EV</code>	3.1	4.7	5.1
<code>Zero</code>	5.9	1.0	0.0
<code>Random</code>	10.9	14.1	-5.0

Table 1: Average NE regret, symmetric profile regret, and symmetric profile payoff of each strategy.

4.5 Discussion

In all of our EGTA experiments, players consistently chose the `Low_Default` strategy in equilibrium. This strategy results in a centralized credit network structure in which a few highly-connected agents facilitate trade among the rest, similar in spirit to the central currency model described by Dandekar *et al.* [2011]. These central agents are attractive targets for credit

allocations for two reasons. First, they have relatively low default risk. Second, given that everyone else is issuing them credit, a link from any one of them creates a path to every other node in the network. Consider a source (yellow) node in Figure 2. Initially it has no credit, and cannot initiate any transactions as a buyer. As soon as any blue node buys from it, however, it obtains an incoming link, which opens up directed paths to purchase from any other node. In other words, the fact that blue nodes are universally trusted makes them central to facilitating trade in the network.

The ability to coordinate on centralized currencies using the `Low.Default` strategy depends on our assumption that agents have a common prior on the default events (i.e., all nodes assign the same default probability to a given node). It remains open for future work to establish whether strategies that coordinate on central currencies will emerge in situations where agents have heterogeneous beliefs about default, or where defaults themselves are conditional on the identity of neighbors.

It is also clear that even in the current setup, there could be other equilibria under some possible settings of our environment parameters. If the cost or probability of defaults were substantially higher, agents would hardly ever have incentive to extend credit and the `Zero` strategy would dominate. If the surplus from transactions were substantially larger, their value might outweigh the various costs to the point where agents might wish to extend credit to everyone to maximize liquidity. Similar effects might also result from varying the length of the simulation.

We plan to experimentally study the setting where, as in the model of Section 3, a network is exogenously given. Since edges in the network represent pre-existing relationships, adjacent nodes can more precisely evaluate each others' default probabilities than can non-neighbors. Thus nodes will be more inclined to extend credit to their neighbors in the graph. This model could also be generalized so that nodes may assign arbitrary default probabilities to each other in a pairwise manner.

5 Conclusion

We report progress on our investigation of strategic issues in the formation of credit networks. Our work to date has proceeded on two tracks, one based on theoretical examination of an analytic model, and the other emphasizing empirical analysis of simulation experiments. Both employ game-theoretic principles, and aim at characterizing the network of credit relationships that can arise from rational and autonomous credit-granting decisions.

The theoretical analysis treats a network contribution game, where granting of credit is a form of contributing to network liquidity. Our model restricts attention to direct bilateral credit transactions, but in return achieves general welfare results with qualitative conditions on the value distributions over transactions. In particular, we identify a class of environments where Nash equilibria and social welfare maximization coincide, which means that self-interested agents can achieve globally desirable outcomes.

The simulation-based analysis covers a more complex transaction environment, where payments may be routed along

paths of credit relationships. The question on this track is about how networks get formed when agents' credit-granting decisions trade off the goals of facilitating profitable transactions and avoiding default risk. We employ empirical game-theoretic methods to identify strategically stable configurations over a set of heuristic strategies. Results from our simulations so far support the notion that if there is a shared prior on default probabilities of nodes, a centralized network emerges where a few trustworthy nodes facilitate trade in the network. It leaves open the question of what kind of network is formed when nodes do not have a common prior on default probabilities.

There is a great deal of further work required to establish our results in more general settings, and answer further strategic questions about formation of credit networks. We plan to continue our investigations on both tracks presented here, and seek ways to shape each track by the results of the other as we proceed.

References

- Elliot Anshelevich and Martin Hoefer. Contribution games in networks. *Algorithmica*, forthcoming.
- Shih-Fen Cheng, Daniel M. Reeves, Yevgeniy Vorobeychik, and Michael P. Wellman. Notes on equilibria in symmetric games. In *AAMAS-04 Workshop on Game-Theoretic and Decision-Theoretic Agents*, New York, 2004.
- Jacomo Corbo, Antoni Calvó-Armengol, and David Parkes. A study of Nash equilibrium in contribution games for peer-to-peer networks. *SIGOPS Operating Systems Review*, 40:61–66, 2006.
- Pranav Dandekar, Ashish Goel, Ramesh Govindan, and Ian Post. Liquidity in credit networks: A little trust goes a long way. In *12th ACM Conference on Electronic Commerce*, San Jose, CA, 2011.
- Dimitri do B. DeFigueiredo and Earl T. Barr. TrustDavis: A non-exploitable online reputation system. In *7th IEEE International Conference on E-Commerce Technology*, pages 274–283, Washington, DC, 2005.
- Alex Fabrikant, Ankur Luthra, Elitza N. Maneva, Christos H. Papadimitriou, and Scott Shenker. On a network creation game. In *22nd ACM Symposium on Principles of Distributed Computing*, pages 347–351, 2003.
- Arpita Ghosh, Mohammad Mahdian, Daniel M. Reeves, David M. Pennock, and Ryan Fugger. Mechanism design on trust networks. In *3rd International Workshop on Internet and Network Economics*, pages 257–268, 2007.
- Matthew O. Jackson and Asher Wolinsky. A strategic model of social and economic networks. *Journal of Economic Theory*, 71(1):44–74, 1996.
- Matthew O. Jackson, Tomás Rodríguez Barraquer, and Xu Tan. Social capital and social quilts: Network patterns of favor exchange. Working paper, Stanford University, 2010.
- Patrick R Jordan. *Practical strategic reasoning with applications in market games*. PhD thesis, University of Michigan, 2010.

- Dean Karlan, Markus Mobius, Tanya Rosenblat, and Adam Szeidl. Trust and social collateral. *Quarterly Journal of Economics*, 124(3):1307–1361, 2009.
- Zhengye Liu, Hao Hu, Yong Liu, Keith W. Ross, Yao Wang, and Markus Mobius. P2P trading in social networks: The value of staying connected. In *29th IEEE International Conference on Computer Communications*, pages 1–9, 2010.
- Alan Mislove, Ansley Post, Peter Druschel, and Krishna P. Gummadi. Ostra: Leveraging trust to thwart unwanted communication. In *5th Usenix Symposium on Networked Systems Design and Implementation*, pages 15–30, San Francisco, 2008.
- Abraham Neyman. Correlated equilibrium and potential games. *International Journal of Game Theory*, 26:223–227, 1997.
- Michael P. Wellman, Daniel M. Reeves, Kevin M. Lochner, Shih-Fen Cheng, and Rahul Suri. Approximate strategic reasoning through hierarchical reduction of large symmetric games. In *20th National Conference on Artificial Intelligence*, pages 502–508, Pittsburgh, 2005.
- Michael P. Wellman. Methods for empirical game-theoretic analysis (extended abstract). In *21st National Conference on Artificial Intelligence*, pages 1552–1555, Boston, 2006.

Dealing with Trust and Reputation in unreliable Multi-agent Trading Environments

Iraklis Tsekourakis and Andreas L. Symeonidis

Centre for Research and Technology Hellas

&

Electrical and Computer Engineering Department, Aristotle University of Thessaloniki
Thessaloniki, Greece

htsekourakis@gmail.com, asymeon@eng.auth.gr

Abstract

In shared competitive environments, where information comes from various sources, agents may interact with each other in a competitive manner in order to achieve their individual goals. Numerous research efforts exist, attempting to define protocols, rules and interfaces for agents to abide by and ensure trustworthy exchange of information. Auction environments and e-commerce platforms are such paradigms, where trust and reputation are vital factors determining agent strategy. And though the process is always secured with a number of safeguards, there is always the issue of unreliability. In this context, the Agent Reputation and Trust (ART) testbed has provided researchers with the ability to test different trust and reputation strategies, in various types of trust/reputation environments. Current work attempts to identify the most viable trust and reputation models stated in the literature, while it further elaborates on the issue by proposing a robust trust and reputation mechanism. This mechanism is incorporated in our agent, *HerculAgent*, and tested in a variety of environments against the top performing agents of the ART competition. The paper provides a thorough analysis of ART, presents *HerculAgent*'s architecture and discuss its performance.

1 Introduction

Agent Technology (AT) is constantly gaining ground in domains where continuous interaction is required. Software Agents may act in uncertain and dynamic environments, adapt – illustrating various levels of autonomy – and collaborate or compete in order to achieve their goals. Examples of such dynamic domains are Peer to Peer (P2P) networks, e-business and m-commerce solutions, autonomic and grid computing, as well as pervasive computing environments [Simon, 1996].

It is more than obvious that interaction may entail malice, with agents (human or software) aiming to promote own interest while at the same time disserving others'. In order to deal with this problem, the concepts of *trust* and *reputation*

(T&R) are employed, providing agents with useful insight on which agents to trust and interact with.

Current work aims to analyze and discuss existing approaches on trust and reputation. Analysis is performed against the Agent Reputation and Trust testbed, a multi-parametric environment designed and developed for testing various trust strategies. Based on the analysis performed, a trust and reputation mechanism is developed and embedded into *HerculAgent* that is benchmarked against the top scoring agents of the ART competition.

The paper is organized as follows: Section 2 discusses state-of-the-art on the available trust and reputation models, while Section 3 provides an overview of ART, discusses the winning T&R strategies and performs a preliminary analysis in order to identify the key factors affecting performance. Section 4 introduces the proposed T&R model and outlines the *HerculAgent* architecture. Finally, Section 5 discusses the performance of the agent on a set of experiments, while Section 6 proposes future directions and concludes the paper.

2 Trust and Reputation Models

There exists extensive literature related to trust and reputation, since it is strongly related to the application domain and the technologies used. Nevertheless, all approaches share a set of common factors, which are discussed within the context of this Section. Additionally, one should keep in mind that current work is focused on the T&R aspects in multi-agent trading environments, thus emphasis is given in that direction.

2.1 Specifying Trust and Reputation

Trust is the fundamental concern in open distributed systems. It lies at the core of all interactions between the entities that have to operate in uncertain and constantly changing environments [Sarvapali et al., 2004]. In case of open multi-agent trading environments, trust pervades multi-agent interactions at all levels. In general, trust models are useful in spotting and marginalizing unreliable/malicious agents, in evaluating the outcome of an interaction and in leading to decisions on trustworthy agents to transact with.

Trust may be conceptualized in the following ways:

- **Individual-level trust**, whereby an agent has a set of beliefs about the honesty or reciprocate nature of the agents it interacts with;
- **System-level trust**, whereby agents operating in an environment are forced to be trustworthy by the rules of encounter (i.e. protocols and mechanisms) that regulate the system.

Should one discuss trust in the broader context where agents may act according to self-interest, and given that system-level trust mechanisms take a *de facto* approach on agent honesty, it is evident that individual-level trust issues are of great importance in contemporary trading environments. Research literature proposes three main approaches for specifying trust: the use of *A priori* evidence, the use of *Experienced* evidence, and the use of *Reputation*.

2.1.1 A priori evidence

A priori evidence is evidence provided by specific protocols, policies and mechanisms which guarantee trust between participants [Huang et al., 2008]. In other words, when an agent acts following the rules that the protocols, policies, or mechanisms dictate ensures that this agent can be trusted.

2.1.2 Experienced evidence

As its name implies, experienced evidence is retrieved by agent interactions. This category is classified into two sub-categories: *direct* experience evidence [Ramchurn et al., 2003] and *witness* evidence [Gomez et al., 2007].

Direct experience is the most relevant and reliable evidence source for trust management. It is the information an agent gains through the direct interactions with its partners. The trust reasoning efficiency of an agent is proportionate to the size of the interaction history saved by the agent. It is, though, disproportionate to the evidential effectiveness.

Witness evidence originates from the interactions of other agents in the community, which in turn may come from direct experience or witness evidence. Thus, the accuracy of evidence is strongly related to the source of evidence; due to its uncertainty, witness evidence is rarely exploited in existing trading environments.

2.1.3 Reputation management

Reputation is the most exploited concept in trust management of multi-agent systems. Though the definition of reputation varies with respect to the context of the domain it is applied, one could argue that reputation is expressed as three levels of rating that may express the trustworthiness of an agent against other agents: *individual ratings*, *collective ratings* and the *rating transmissions*. In terms of rating, the techniques that manage the ratings could be divided to *rating retrieval* techniques, and *rating aggregation* techniques.

Rating retrieval is applied on distributed trading environments, where the topology is not known beforehand and network analysis techniques are employed in order to retrieve ratings [Fullam and Barber, 2006], [Sabater and Sierra, 2002]. Having retrieved ratings, rating aggregation is performed in order to calculate reputation and define trust.

In literature, rating aggregation may be performed in a number of manners, ranging from naive to more sophisticated ones.

2.2 Efficiency of Trust and Reputation Models

A successful trust and reputation model depends on the type of evidence the model provides to agents, the techniques used to get the above evidence, and the way an agent handles such evidence to extract trustworthiness for others. Efficiency of a T&R model is defined with respect to the following axes [Fullam et al. 2005]:

- **Accuracy.** T&R models must provide good prediction on another agent's future behavior [Fullam, 2003].
- **Adaptivity.** T&R models must be able to adapt in order to accommodate dynamic trustworthiness characteristics of other agents [Fullam and Barber, 2005].
- **Quick Convergence.** T&R modeling algorithms must quickly generate new models when unknown agents enter the system [Ding et al, 2005].
- **Multidimensionality.** T&R models must differentiate between another agent's varied trustworthiness characteristics across multiple categories [Muller et al, 2003].
- **Efficiency.** T&R algorithms must generate models with minimal computational cost and in minimal time.

3 The ART testbed

3.1 The ART scenario and architecture

As already discussed, a variety of approaches exist, aiming to model trust and reputation in multi-agent systems. The Agent Reputation and Trust (ART) testbed [URL 1] provides an ideal framework for benchmarking different T&R strategies.

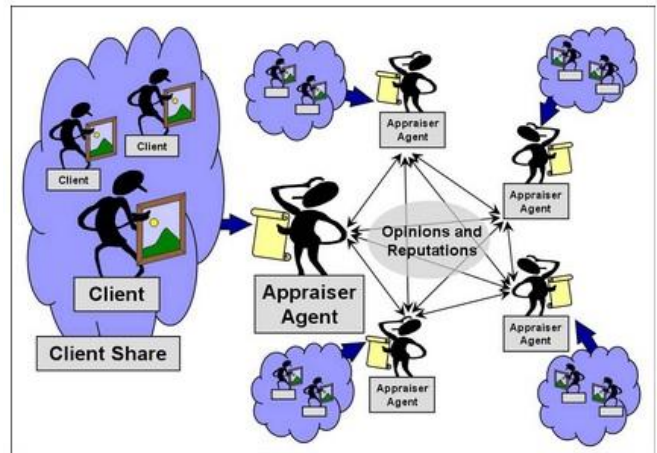


Figure 1. ART entities and their interactions.

Within the context of ART, each agent represent an art appraiser, competing against all other agents (appraisers) in the system. Clients (handled by the ART server) request

appraisals for paintings from different eras. In case an appraiser is an expert on paintings of the specific era, it is capable for providing an accurate appraisal, thus satisfying the client that will buy the painting and pay the appraiser. In case the agent is not an expert on paintings of the era, it may request – paying a fee – an evaluation (defined as opinion) by other appraisers. Appraisers may also transact with each other on reputation information on other appraisers. Based on their T&R strategy, agents must decide when and from whom to request opinions and reputation information, in order to generate accurate appraisals for clients. The more accurate the appraisals, the more the clients attracted and profit for the appraiser. Winner agent is declared the one with the highest bank account balance. Figure 1 illustrates the possible interactions and the type of information exchanged between appraisers and clients. More information on ART can be found at [URL 1].

The ART testbed comprises four basic modules [Fullam et al., 2005b]:

- The *Simulation Engine*, which is responsible for generating controlled T&R environments by enforcing user-defined parameters.
- The *ART Database*, which stores all game information for reporting and later retrieval.
- The *ART GUI*, providing access to online game monitoring and result visualization.
- The *Agent Skeleton*, an agent wrapper for researchers to embed their T&R strategy, while ensuring unflustered communication with the other ART entities.

3.2 ART T&R modeling and State-of-the-art

Within the context of ART, an agent T&R strategy should span across three axes: (i) modeling of the other agents (environment), (ii) modeling request and, (iii) modeling response. The response strategy directly affects its’ reputation, while the request and response strategies have an effect on environment modeling [Kafali and Yolum, 2006].

The basic challenges in building an efficient T&R strategy have been identified by Costa et al. [2008] and are: (i) the start game opinion requests, (ii) the identification of trustworthy agents, (iii) the appraisal definition policy and complexity, (iv) the reputation definition, request and response policies and, finally, the aggregation of all the above. Researchers may focus on one or more aspects of the strategy, in order to build an efficient agent.

May one go through related literature, one may identify several approaches that perform ART T&R modeling. Nevertheless, two are the ones that stand out: UNO and IAM.

3.2.1 UNO

Probably the most successful and efficient ART T&R model. Murillo and Munoz [2008] focused on the request and response aspects of the T&R model. In both cases they exploit knowledge the agent builds on the other agents, based on:

- the error of own appraisals

- if an agent has responded to an UNO request and,
- the total number of requests an agent has made to UNO.

It is worth mentioning that UNO does not employ a reputation mechanism to extract trust values, since the UNO team considered that the number of participating agents is too small for generating trustworthy reputation estimations. Thus, they selected to work directly with the real appraisal error of each agent.

3.2.2 IAM

IAM is another successful ART agent paradigm. Teacy et al. [2007] describe IAM behavior as *Intelligent* (using statistical models for opponent modeling), *Abstemious* (spending its budget parsimoniously based on its trust model) and *Moral* (providing fair and honest feedback to those that request it). IAM decides based on the following information:

- Appraisal responses from the Simulator Engine and other appraisers
- Information on the behavior of other agents (e.g. reputation values).

The trust model of IAM comprises three parts: (a) the *lie detector*, which identifies malicious agents, (b) the *variation appraiser*, which estimates the variation of appraisal errors of the other agents and (c) the *weight estimator of the most accurate agents*. In contrast to UNO, IAM employs reputation in order to build its T&R model.

3.3 Preliminary analysis of the ART environment

In order to identify the factors that mostly affect appraiser performance, we performed an extensive set of experiments, where different game parameters and simple policies were tested. We employed the *SimpleAgent* (provided by the ART framework) and gradually tested it against the *TestAgent* (*SimpleAgent* equipped with a simple policy), the *CheatingAgent* and *HonestAgent*, as well as the agents that participated in the 2008 ART competition. In the latter experiments, more elaborate policies were followed, based on the analysis performed. In all cases the game parameters were the same; these are defined in Table 1.

Table 1. Game parameters during experimentation

Time epochs	50
Eras	10
Average number of clients per appraiser	20
Client fee	100
Appraisal cost	10
Certainty cost	1
Reputation cost	0.1
Appraisal messages	2
Certainty messages	20
Denial of personal opinion	True
Variable eras	2
Expertise value change	0.05

The types of policies investigated are discussed in Table 2, where the last column defines the number of discrete

steps selected for each of the T&R factors. It should be mentioned that the following results are aggregates following monte carlo analysis.

Table 2. The various policies applied

Policy	T&R factor	Steps
Pol-1	Reputation definition policy	7
Pol-2	Reputation request policy (reputation only)	5
Pol-3	Reputation response policy	3
Pol-4	Honesty policy (based on agent trustworthiness)	3
Pol-5	Reputation request policy (reputation and certainty)	3
Pol-6	Appraisal cost for trustworthy agents	5
Pol-7	Appraisal cost for unreliable agents	5
Pol-8	Trustworthiness with respect to appraisal cost	3
Pol-9	Optimal reputation definition policy (Pol-1) Optimal trustworthiness policy (Pol-4) Honest response to trustworthy agents Dishonest response to unreliable agents	4
Pol-10	Optimal reputation definition policy (Pol-1) Honest response to trustworthy agents Dishonest response to unreliable agents Optimal appraisal cost policy (Pol-6)	3
Pol-11	Optimal reputation definition policy (Pol-1) Optimal reputation response policy (Pol-3) Optimal appraisal cost policy (Pol-6) Honest response to trustworthy agents Dishonest response to unreliable agents	3
Pol-12	Optimal reputation definition policy (Pol-1) Optimal reputation response policy (Pol-3) Optimal appraisal cost policy (Pol-6, Pol-7) Honest response to trustworthy agents Dishonest response to unreliable agents	3

3.3.1 TestAgent vs. SimpleAgent

An extensive set of tests was performed on the simple policies applied (Pol.1 – Pol.8), in order to identify which of the T&R factors affect appraiser performance. Subfigures 2.1 – 2.8 illustrate the performance of the competing agents in various configurations (omitted due to space limitations). Block 1 denotes the set of basic rules identified and the optimal values of the most important factors:

BLOCK 1. BASIC RULES IDENTIFIED

Rule-1. IF $ME > 0.5$ THEN $rep = rep - 0.02$ ELSE $rep = rep + 0.04$

Rule-2. IF $rep_{Appraiser} > 0.5$ THEN it is considered **trustworthy**

Rule-3. IF an appraiser is trustworthy
THEN provide accurate opinions
ELSE provide falsified opinions

Rule-4. IF an appraiser is trustworthy
THEN pay $0.7 * AppCost$ to get an appraisal
ELSE pay $0.15 * AppCost$

3.3.2 TestAgent vs. HonestAgent and CheatingAgent

The ART testbed provides two more agents for benchmarking: the *CheatingAgent* and *HonestAgent*. *TestAgent* equipped with the knowledge base generated during the first experimentation stage (Pol.1 – Pol.8), was benchmarked against these agents. Subfigures 2.9 – 2.11 illustrate the per-

formance of the competing agents in various configurations. Apart all other observations, one should also point out that in some cases, *SimpleAgent* stills outperforms *TestAgent*, since the strategy the latter follows is oriented towards more complex strategies.

3.3.3 TestAgent vs. ART 2008 Winning Agents

Finally, *TestAgent* was benchmarked against the top performing agents of the ART 2008 competition. Subfigure 2.12 illustrates the performance of the competing agents in various configurations. At this point *TestAgent* outperforms *SimpleAgent*, since the latter cannot cope with the complexity of the competitors' strategies. Nevertheless, it is obvious that the approach *TestAgent* follows still lacks dynamicity and adjustability.

A number of useful observations were made through the analysis performed. First of all, computing reputation based on direct interactions with other agents proved more efficient than using information from reputation responses. Additionally, experimentation dictated that neither honesty, nor unreliability work alone. Competitors should be forced to play fair, so as to be rewarded with trust (or penalized with unreliability). Finally, the ranges of metric values were identified, where our agent increased efficiency. Outside of these ranges agent performance decreases. Based on these observations, *HerculAgent* was developed.

4 HerculAgent architecture

HerculAgent follows a modular architecture, so as to meet each of the diverse needs imposed by the ART framework (Figure 3). Its behavior is expressed through nine strategy functions, which are aggregated into three behavior modules implementing three protocols: the *reputation protocol*, the *certainty protocol* and the *opinion protocol*.

4.2 HerculAgent Protocols

4.1 Reputation Protocol

The reputation protocol manages the reputation values Rep_{ijt} ($i: Agent_i, j: Era_j, t: Epoch_t$), primarily based on previous direct interactions, and secondarily on indirect sources such as observation information. The *reputation* module actually deliberates on the reward or admonition strategy to follow, based on the appraisal estimates provided by other appraisers in the past. It also defines the number of reputation requests *HerculAgent* will make and the agents to request reputation information from. Finally, the *reputation* module determines the response strategy to reputation requests the agent receives from other agents.

4.1.2 Certainty Protocol

The certainty protocol manages the certainty values Cer_{ijt} ($i: Agent_i, j: Era_j, t: Epoch_t$) and refers to the confidence an agent has on the reputation of other appraisers. The *certainty* module implements the strategy *HerculAgent* has with respect to which agents to ask for their certainty values.

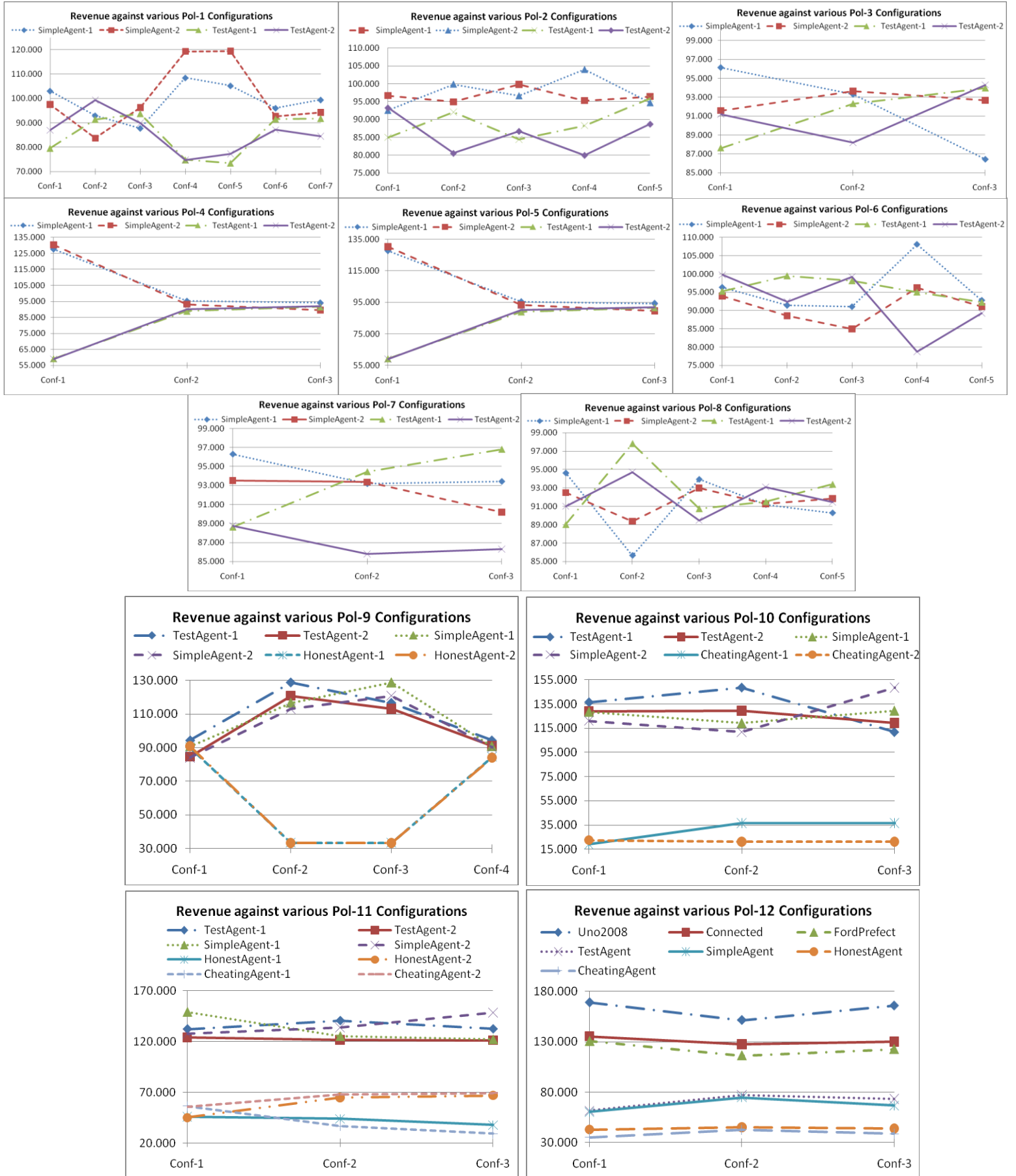


Figure 2. Appraiser Revenue with respect to the various Policies applied and the configuration settings selected

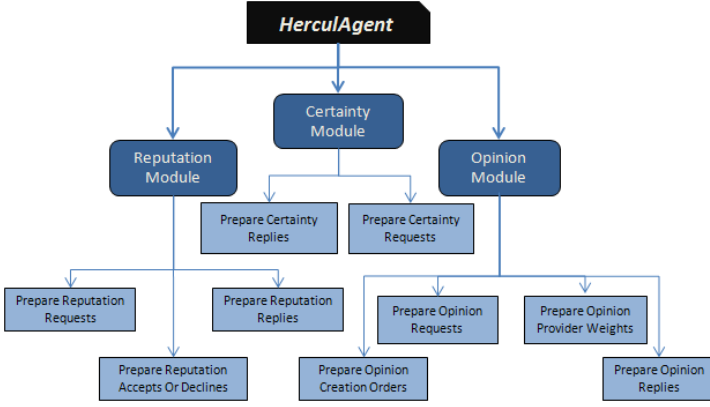


Figure 3. The HercuAgent architecture

4.1.3 Opinion Protocol

Finally, the opinion protocol is the core business protocol of the agent, responsible for issuing accurate painting appraisals/opinions. The *opinion* module selects the agents to trust and request their opinion, while it also builds its own personal opinions base to respond to competition agents. It contemplates the strategy that the agent uses to define the combination of the other agents' appraisals to create its own final appraisal and finally send it to the client.

4.2 HercuAgent Metrics

HercuAgent employs four metrics in order to design and develop its T&R strategy:

- **Mean error ($ME(i,j,t)$)** is the weighted average of the appraisal relative errors an agent makes and varies for each era. ME depends on past interactions and is defined as:

$$ME(i, j, t) = \frac{((ME(i, j, t-1) \cdot ErC(i, j, t-1) + Er(i, j, t-1)))}{(ErC(i, j, t-1) + 1)},$$

where $Er(i,j,t)$, is the relative appraisal error $Agent_i$ makes for a painting of Era_j at $Epoch_t$ of the game. $ErC(i,j,t)$ is the respective number of the errors.

- **Reputation ($Rep(i,j,t)$)** expresses the level of trustworthiness of $Agent_i$ for a painting of Era_j at $Epoch_t$ of the game. It ranges in the $[0,1]$ interval.
- **Certainty ($Cer(i,j,t)$)** indicates the certainty that an $Agent_i$ claims on its appraisal values for a painting of Era_j at epoch $Epoch_t$ of the game. It ranges in the $[0,1]$ interval.
- **Self Confidence ($SC(i,j,t)$)** expresses the certainty *HercuAgent* has on the other appraiser agents, as defined through past interactions. It denotes the possibility that the $Rep(i,j,t)$ value computed for an appraiser is similar to its $Cer(i,j,t)$ value. Block-2 denotes the pseudocode implementation of SC:

```

BLOCK 2. calculates() PSEUDOCODE
FUNCTION calculateSC(Agent i, Era j, Epoch t){
L1: tempConf <- selfConfidence(Agenti(Appraisal(Eraj)));
L2: IF (tempConf EQUALS 0) THEN tempConf = α
    ELSE tempConf = tempConf + ((1-tempConf) * β)1;
L3: selfConfidence(Agenti(Appraisal(Eraj))) <- tempConf;
} 1 After experimentation, α = 0.01, β = 0.005

```

Metrics are continuously calculated for each agent, era and epoch.

4.3 Dynamic behavior adaptation

HercuAgent employs two methods in order to adapt its behavior and strategy with respect to the data collected throughout the game:

- **setRepLimit()**, which adapts, for each $Agent_i$ for paintings of Era_j at each $Epoch_t$ the minimum $Rep(i,j,t)$ value $Agent_i$ has to meet to be trusted.
- **setErrorsLimit()**, which defines, for each $Agent_i$, for paintings of Era_j at each $Epoch_t$ the error limit that is acceptable for an appraiser agent.

Block-3 denotes the pseudocode implementation of **setErrorsLimit()**. Function **setRepLimit()** is deployed in a similar manner.

BLOCK 3. setErrorsLimit() PSEUDOCODE

```

FUNCTION setErrorsLimit(Agent i, Era j, Epoch t){
L1: FOR (1 TO numberOfEras){
L2:   min <- 1;
L3:   max <- 0;
L4:   FOR (1 TO numberOfAgents){
L5:     IF (errors(Agenti, Eraj) < min
L6:       THEN min <- errors(Agenti, Eraj);
L7:     IF errors(Agenti, Eraj) > max
L8:       THEN max <- errors(Agenti, Eraj);
L9:   }
ErrorsLimit <- min + γ*max2;
} 1 After experimentation, γ = 0.2

```

4.3 HercuAgent Behavior

The core T&R model of *HercuAgent* focuses on trust, and the degree of trustworthiness we show to opponents. In order to decide, three behaviors were implemented: (i) the *typical* behavior (our basic strategy), (ii) the *optimistic* behavior, where opponents are given more “credit” and, (iii) the *pessimistic* behavior, where opponents are given less “credit”. In a similar manner, three behaviors were implemented in order to calculate the weighted average of the final appraisal: the *typical*, *aggressive*, and *submissive* behaviors. All behaviors are specified in the respective *HercuAgent.conf* file and behavior changes dynamically (upon game initiation).

5 Experiments

A number of experiments were performed with *HercuAgent* participating in all agent scenarios, as defined in Section 3. Various strategies were applied and interesting conclusions were drawn. The following results are aggregates following monte carlo analysis.

At first, *HercuAgent* was tested against the naïve set of agents that were tested in the preliminary phase, and easily outperformed them. Figure 4 illustrates agent revenue (Bank balance) of an indicative game, as depicted by the ART Light Game Monitor Interface.

Consequently, *HercuAgent* was tested against the top performing agents of the ART 2008 competition. Figure 5

illustrates agent revenue (Bank balance) of an indicative game, while Figure 6 presents the aggregate results with respect to the different *HerculAgent* behaviors.

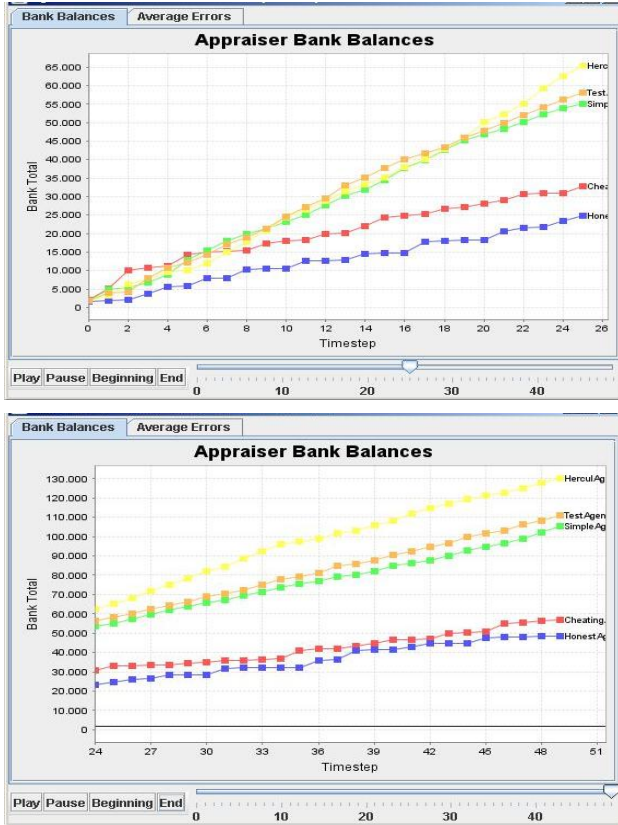


Figure 4. *HerculAgent* against the naive set of agents

Through the numerous experiments performed in order to compare our strategy against the winning agents of the ART 2008 competition, we observe that our results are satisfactory but could be further improved. *HerculAgent* often succeeded in finished second third, nevertheless never succeeded in beating *Uno*. The strategy of *HerculAgent* did not perform adequately at the first half of the game (start game effect). This can be justified based on the fact that *HerculAgent* computes trust mainly on information received through direct interactions. Thus, during the initial epochs there is not enough information available for reasoning. Whenever accurate initial appraisals are performed, the agent performs very well. In all cases, though, in the second half of the game, the bank total of *HerculAgent* improves significantly, at a rate even greater than the winner agent *Uno*.

6 Conclusions - Future Work

Current work discusses *HerculAgent*, an agent designed and developed for the analysis of various T&R models in dynamic trading environments. The ART platform was selected for experimentation, given that it provides a unique testbed for testing various game parameters. Preliminary analysis indicated the basic factors affecting performance, and a set of rules of thumb were identified, which were later embedded in our agent model.

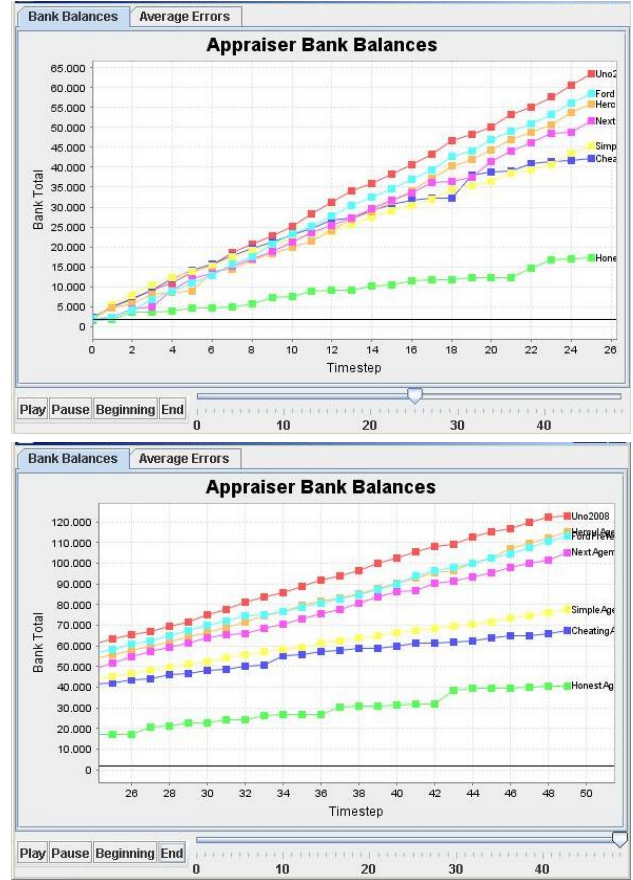


Figure 5. *HerculAgent* against the ART 2008 top performing agents

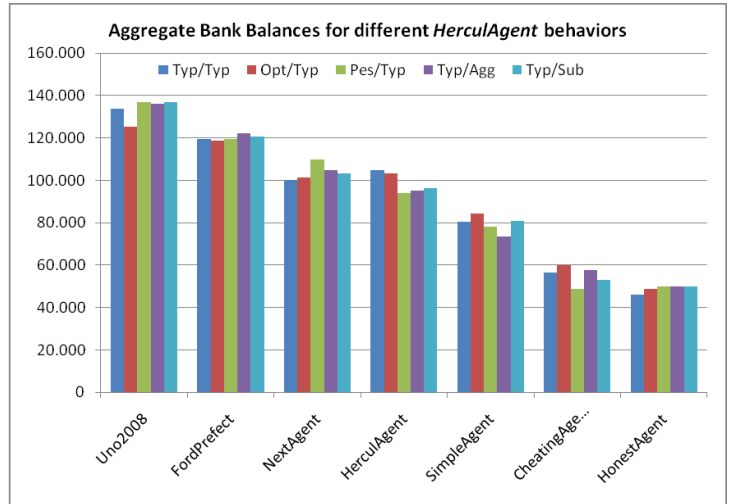


Figure 6. Aggregate performance of *HerculAgent* against the ART 2008 top performing agents

Results show the strong points and drawbacks of *HerculAgent*. It is worth noticing that, through direct interactions, trust and reputation are extracted accurately in contrast to

the use of indirect interactions. Main goal of future work are the development of quick and accurate trust extraction algorithms based on self confidence, the development of an off-line mechanism that exploits regression techniques for estimating reputation values, and the improvement of the agent behavior during the start-game effect.

References

- [Ding et al, 2005] L. Ding, P. Kolari, S.Ganjugunte, T. Finin, and A. Joshi. On Modeling and Evaluating Trust Networks Inference. 2005. pp 21-32.
- [Fullam, 2003] K. Fullam. An Expressive Belief Revision Framework Based on Information Valuation. *Master's thesis*, Dept. of EE, U. Texas (Austin), 2003.
- [Fullam and Barber, 2005] K. Fullam and K. Suzanne Barber. A Temporal Policy for Trusting Information. *Trusting Agents for Trusting Electronic Societies. Lecture Notes in Computer Science*, 2005.
- [Fullam and Barber, 2006] K. Fullam and K. S. Barber. Learning trust strategies in reputation exchange networks. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems (AAMAS '06)*, P. Stone and G. Weiss, Eds. New York: ACM Press. pp. 1241-1248, 2006.
- [Fullam et al. 2005] K. Fullam, T. Klos, G. Muller, J. Sabater, A. Schlosser, Z. Topol, S. Barber, J. Rosenschein, L.Vercouter and M.Voss. A Specification of the Agent Reputation and Trust (ART) Testbed: Experimentation and Competition for Trust in Agent Societies. *Proceedings of AAMAS 2005*.
- [Fullam et al., 2005b] K. Fullam, T.B. Klos, G. Muller, J. Sabater, Z. Topol, K.S. Barber. J.S. Rosenschein, and L. Vercouter. The Agent Reputation and Trust (ART) Testbed Architecture. *Proceedings of AAMAS 2005*.
- [Gomez et al., 2007] M. Gomez, J. Carbo and C. Benac Earle. An Anticipatory Trust Model for Open Distributed Systems. In *Lecture Notes in Artificial Intelligence*, 2007.
- [Huang et al., 2008] Hongbing Huang, Guiming Zhu, and Shiyao Jin. Revisiting Trust and Reputation in Multi-Agent Systems. In the *2008 ISECS International Colloquium on Computing, Communication, Control, and Management*. 2008.
- [Kafali and Yolum, 2006] O. Kafali and P. Yolum. Trust Strategies for ART Testbed. In the *International Workshop on Trust in Agent Societies*. 2006.
- [Muller et al, 2003] G. Muller, L. Vercouter, and O. Boissier. Towards a general definition of trust and its application to openness in MAS. *Proc. of the AAMAS-2003 Workshop on Deception, Fraud and Trust*, 2003.
- [Murillo and Munoz, 2008] Murillo J. & Munoz V., 2008. Agent UNO: Winner in the 2nd Spanish ART Competition. *Inteligencia Artificial*. 2008.
- [Ramchurn et al., 2003] S.D. Ramchurn, C. Sierra, G.L., and N.R. Jennings. A computational trust model for multi-agent interactions based on Confidence and reputation. In the *6th International Workshop of Deception, Fraud and Trust in Agent Societies*, 2003.
- [Sabater and Sierra, 2002] Sabater J. & Sierra C., 2002. Reputation and social network analysis in multi-agent systems. In *Proceedings of the first international joint conference on Autonomous agents and multi-agent systems (AAMAS '02)*, New York: ACM Press. pp. 475-482.
- [Sarvapali et al., 2004] Sarvapali D. Ramchurn, Dong Huynh & Jennings NR. Trust in Multi-Agent Systems. *The Knowledge Engineering Review*. 2004.
- [Sen et al., 2006] S. Sen, I. Goswami, and S. Airiau. Expertise and Trustbased Formation of Effective Coalitions: An Evaluation of the (ART) Testbed. In *Proc. Trust Workshop at AAMAS*, 2006.
- [Simon, 1996] Simon H. The Sciences of the Artificial. *MIT Press*, MA, Cambridge. 1996.
- [Teacy et al., 2007] Teacy, W. T. L., Huynh, T. D., Dash, R. K., Jennings, N. R., Luck, M. & Patel, J. The ART of IAM: The Winning Strategy for the 2006 Competition. In: *The 10th International Workshop on Trust in Agent Societies*, May 2007, Honolulu, Hawaii, USA. pp.
- [URL 1] <http://megatron.iia.csic.es/art-testbed/>

Improving Prediction in TAC SCM by Integrating Multivariate and Temporal Aspects via PLS Regression

William Groves and Maria Gini

University of Minnesota

Department of Computer Science and Engineering, Minneapolis, MN 55455

{groves, gini}@cs.umn.edu

Abstract

We address the construction of a prediction model from data available in a complex environment. We first present a data extraction method that is able to leverage information contained in the movements of all variables in recent observations. This improved data extraction is then used with a common multivariate regression technique: Partial Least Squares (PLS) regression. We experimentally validate this combined data extraction and modeling with data from a competitive multi-agent supply chain setting, the Trading Agent Competition for Supply Chain Management (TAC SCM). Our method achieves competitive (and often superior) performance compared to the state-of-the-art domain-specific prediction techniques used in the 2008 Prediction Challenge competition.

1 Introduction

The basic purpose of prediction in decision making scenarios is to provide information about the future that is relevant to decisions that must be made now. In a supply chain scenario, prediction is generally concerned with providing information about future costs, prices, and demand so that decisions can be made today that maximize utility. Other aspects such as limited information, informational delays, changing conditions, and observation noise can add complexity and make prediction even more difficult. The Trading Agent Competition for Supply Chain Management (TAC SCM) is a market simulation game that possesses all these challenges.

TAC SCM is a complex, heterogeneous-agent supply chain game that is designed to simulate an oligopoly market of competing computer manufacturing agents who must autonomously purchase component parts, manufacture computers, and sell the finished computers to end customers over a simulated product lifetime of a year. Each run of the simulation, which takes approximately one hour in real-time, is unique because the market's behavior will vary both due to changes in the underlying market conditions (available component supply and customer demand), and due to changes in the behavior of individual agents. This competition, which has been held annually since 2003, has attracted many participants from around the world [Collins et al., 2006].

For the purposes of this paper, we are primarily concerned with the price prediction aspects of the competition. A sub-competition called the "Prediction Challenge" was initiated in 2007 to isolate this aspect of the simulation for study [Pardoe and Stone, 2008]. This subcompetition facilitated direct comparisons of prediction performance on four prediction types: (1) current product prices, (2) future product prices, (3) current component prices, and (4) future component prices. "Current" refers to predictions about prices revealed to an agent on the next day; "future" refers to predictions about prices revealed to an agent an additional twenty days in the future from the current day. For each prediction class, we show how to generate an optimal feature set and the prediction results of PLS regression using these features.

The challenge of predicting in this environment is that many of the variables have relationships to other variables that are not easily quantified. Economic theory suggests that there is a direct relationship between the cost of parts and the cost of finished goods, but, in a situation where parts are shared across many products, the relationships are dynamic. Instead of modeling these relationships individually, we use partial least squares (PLS) regression, which is able to implicitly determine correlations between input variables based on historical data. To effectively model these relationships, a corpus of representative historical data is required to calibrate the model. The observations used to construct the training set features should be representative of the range of values the model will need to predict over. In a multi-agent setting, this means that the training data should be generated by a similar set of opponents as when predictions are to be made.

We make two novel contributions: (1) we include time-delayed observations as additional elements in the feature vector, in addition to the most recent observed value of each variable, and (2) we segment the features hierarchically so that searching for a higher-performance combination of time-delayed features is tractable and efficient. Despite the fact our method uses very little domain-knowledge, we achieved competitive (and often superior) performance compared to the domain-specific prediction techniques in the 2008 Prediction Challenge.

The paper is structured as follows: in the next section, we present related work relevant to PLS regression and previous prediction approaches in TAC SCM. In Section 3, our data extraction and prediction methodology is outlined including

how the feature vector is augmented to include data from past observations. Our prediction results are presented and compared with the state-of-the-art methods in Section 5. Section 6 concludes with a discussion of future work.

2 Background and Related Work

The TAC SCM competition as well as other autonomous agent competitions involving travel brokering [Stone and Greenwald, 2005] and stock trading (Penn Lehman Trading Competition) [Kearns and Ortiz, 2003] require agents to do significant forecasting to achieve consistent, strong performance. The methods that employ information learned from historical data are the most effective.

TAC SCM: Current and Future Product

The prediction methods employed by top performers in the TAC SCM competition involve a combination of offline training on historical data for computation of latent (or not always visible) information. The techniques used in each of the four prediction classes will be discussed in turn. In the context of the TAC SCM Prediction Challenge, it is required to predict for the next day and 20 day ahead sale price of each product type. The TacTex agent makes next day product price predictions using a particle filter to compute the lowest price offered by any agent in response to each request [Pardoe and Stone, 2007]. The Deep Maize agent uses a k -nearest neighbor technique over a large set of historical data. The feature vector, over which similarity is computed, includes current day estimates of supplier capacity (which features prominently in the pricing equation), today’s observed customer demand, and recently observed computer prices [Kiekintveld et al., 2009]. This method explicitly limits the contribution of individual games by using only the most similar observation from each game in the data set. The CMieux agent uses a decision tree variant called a distribution tree that tracks several distinct distributions of prices and chooses the most relevant normal distribution to use based on publicly known attributes of the price to estimate [Benisch et al., 2009]. A classification-based approach is used in [Ketter et al., 2009] involving the clustering of observed market behaviors into economic regimes to build both improved current and future product price distribution models. Overall, there are significant tactical reasons to estimate next day prices due to the need to assign a specific bid price to each customer request.

For 20 day ahead sales price predictions (“future product”), most agents report to use methods similar to their next day approaches. The TacTex agent uses the result of its next day particle filter and augments this value by computing an adjustment for 20 day ahead predictions using an additive regression model. The additive regression model estimates the difference between the next day price and the 20 day ahead price and is derived from a set of over 31 visible features (16 product prices, 3 demand signals, and 12 unique component prices) from the current day’s observations. This technique is most similar to the PLS regression method discussed in Section 3 but differs by not using information observed prior to the current day. Deep Maize estimates future product prices using the same k -nearest neighbor used in next day predictions but instead trained on 20 day ahead observations.

Some agents may not be estimating future product prices (20 day ahead predictions) at all because it is not a tactically necessary prediction (agents are not required to compute this value when operating in this environment). Instead, this prediction class may have mainly strategic “long-term” uses for planning long lead-time procurement.

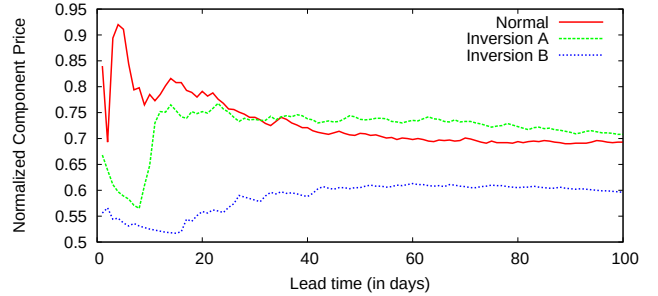


Figure 1: Three snapshots of the price vs. lead-time relationship for a single component. These relationships will change daily both due to stochastic elements and agent behavior.

Component Market Current and Future

Component pricing is significantly more complex due to the need to consider lead time. When an agent makes a request for component parts it must also specify a delivery lead-time. This is critical because the prices quoted vary significantly over the range of possible lead times. Figure 1 shows three example price vs. lead-time plots taken in an individual TAC SCM game and motivates the need for agents to track component prices also in terms of lead-time. The general trend is for prices to be highest in the short term (series “Normal”), but there are instances (series “Inversion” A & B) where short term prices are, in fact, lower. Interestingly, the lead-time aspect of product prices is not explicitly considered by many TAC SCM agents.

TacTex predicts next day component prices with a domain-specific approach by estimating the available production capacity of each supplier. This is done using knowledge of how component prices are calculated by the game server. The fraction of unallocated supplier capacity between the request date and the delivery date is directly proportional to the quoted per unit cost for each request. This value can be estimated by observing differences in the recent quoted prices. This approach cannot be generalized to other domains. For future component requests (20 days ahead), TacTex agent again employs additive regression to learn an adjustment between the current day’s price and the future price (20 day ahead). Their algorithm is otherwise unchanged from the next day method.

Deep Maize uses a linear interpolation over recent samples to estimate the component price for a specific due date. An adjustment to the linear interpolation is computed from a reduced error pruning (REP) tree, a decision tree variant. The REP tree offset values are trained offline using historical data of next day predictions. This same method is used for 20 day ahead predictions [Kiekintveld et al., 2009].

The CMieux agent uses a k -nearest neighbor approach to estimate next day component costs. The contribution of the k -

nearest neighbors is averaged using inverse distance weighting, and distance refers to the temporal difference between the due date requests [Benisch et al., 2009].

Multivariate Techniques

The literature contains several classes of methods that systematically leverage information from multiple related time series including vector autoregressive moving average (ARMA) and multivariate regression methods.

Vector ARMA is the multivariate analog of the ARMA time-series predictor presented in [Box, Pelham, and Jenkins, 1994; Tiao and Box, 1981]. Vector ARMA is applied in [Olason and Watt, 1986] using the MISO TFN algorithm (Multiple Input Single Output Transfer Function-Noise model) to improve river flow predictions on a network of hydropower generating stations. Given daily flow data from 3 upstream and one downstream reservoirs, the authors developed an improved flow prediction model for both next-day and two day ahead predictions. The improvement was made using three years of daily flow measurements from all four sites and reduced standard error over the conventional approach of using separate time-series ARMA models for each site.

The vector ARMA also appears to be effective in domains with a small sample size. Disadvantages of vector ARMA stem from its sensitivity to collinearity in the input variables; the examples from the literature use only a small number (3-5) of input variables.

The family of methods, multiple linear regression (MLR), principal component regression (PCR) and PLS regression are techniques used with significant success in chemometrics, social science, and econometrics [Martens and Næs, 1992]. PLS regression is found to be particularly successful on problems having a large number of highly collinear variables and a small number of samples [Wold, Martens, and Wold, 1983].

The behavior of PLS, MLR, and PCR regression techniques differ from the time-series based techniques due to the absence of a “memory” component. Time series models like ARMA can be affected by outliers in the input data, and the effect of an outlier can persist in the output prediction long after the outlier was first observed [Hillmer, Larcker, and Schroeder, 1983]. This is in contrast to the regression techniques: input observations that are outliers will only affect the output variable when the outlier is still in the input set. In adversarial settings like TAC SCM, where opponents have the ability to temporarily affect the input data to a prediction computation, the long term effect of outliers in time-series predictors is particularly undesirable.

3 Our Approach

The prediction model that PLS regression computes for a variable y is a weighted linear function in terms of the feature values $x_i, i = 1..m$ (where m is the number of features). Mathematically, the prediction model for a variable y can be expressed as¹: $y = \hat{b}_0 + \hat{b}_1x_1 + \hat{b}_2x_2 + \dots + \hat{b}_mx_m$ where $\hat{b}_i, i \in 0..m$ are the regression coefficients computed in the

¹This is a parsimonious expression of the model. In practice, it may be useful to compute y values in a two step process employing, first, the dimensionality reduction and, second, the regression.

model calibration stage. This form of linear model is suitable for economic modeling and is discussed next.

3.1 A simple model

In an economic supply chain context, this simple linear model maps naturally into the domain: the long-run price of a product should be equivalent to a linear sum of the costs of its constituent parts plus the profit margin taken by the manufacturer. The cost of a component could similarly be computed from this ideal model: the long-run cost of some component should be equivalent to a fraction of the prices of products that use the component.

The ideal model presented here, while intuitively attractive, is not sufficient for modeling prices in TAC SCM for several reasons. First, the model does not directly address lead time effects². Also, particularly in the component market, prices vary significantly by lead time: longer lead time requests typically have lower cost. This mismatch between possible lead times in the two markets drives the need for agents to develop mechanisms for coordination, dynamic planning, and prediction. It is for this reason that the range of possible features to include in our model incorporates price observations for multiple component lead times.

Second, the model does not address trends that can be anticipated by observing changes in price over time. Observation of the prices of an individual product over several past time steps ($p_t, p_{t-1}, p_{t-2}, \dots$) provide information to form a prediction about the next value (p_{t+1}). This is equivalent to a univariate time series prediction. Information from previous time steps is included in our model.

Third, visible non-price information about the environment will also have an effect on prices. For instance, the current bank interest rate (the cost of borrowing), and the current storage cost (the cost of holding inventory) will both have effects on price. This is also included in the model.

Fourth, information about the current market situation also has an effect on price. Information about overall aggregate demand is available to the agents by observing the number of products requested by customers each day. This information can also be included in the feature vector. In keeping with the domain-agnostic nature of our approach, we assume a linear relationship for the non-price features as well.

Including all observable information that could have an effect on price into the feature vector is an obvious approach. But in practice, this is impractical. First, as the number of features rises significantly, the effectiveness of a prediction algorithm is likely to degrade. Second, there is almost no limit to the number of possible features that can be added. The inclusion of irrelevant (or low informational value) features is a problem that we address in the next section.

²In both the component and product markets, agents must commit to buy components or sell products far in advance of the delivery date. In the component market, delivery lead times can be between 1 and 200 days into the future. When an agent makes a component request, it must also specify the delivery date for the request. In the product market, delivery lead times are between 3 and 12 days into the future. The delivery date is specified in the request from the customer, and the agent with the winning bid must honor it or a significant financial per-unit late fee is imposed.

Feature Type		Count	Feature
1	Game Instance	2	– storage cost – interest rate
2	Daily Demand	3	– low market – medium market – high market
3	Price for products	16	– SKU 1 ... – SKU 16
4	Price for parts at LT2 ^a	16	– comp100sup1lt2 ^b – comp101sup1lt2 – comp110sup2lt2 – comp111sup2lt2 ...
	Price for parts at LT6	16	– comp100sup1lt6 – comp101sup1lt6 ...
	Price for parts at LT10	16	...
	Price for parts at LT20	16	...
	Price for parts at LT30	16	...
All Features		101	

^aLT2 denotes a delivery lead time of 2 days from the order date.

^bPrice of component 100 from supplier 1 with a delivery lead-time of 2 days.

Table 1: List of all computed features available to an agent.

3.2 Input Feature Computation

We now illustrate how several distinct types of data available are aggregated into a feature vector of consistent size. Data observable from an individual agent’s perspective consists of four feature types:

1. *Game instance features* include values that remain invariant throughout the simulation instance, such as the bank interest rate and storage cost.
2. *Daily market segment demand* is computed from the total number of product RFQs in each of the three market segment on each day.
3. *Daily price observations for products* are the mean sale prices for each of the 16 product types, as observed from the agent. Days with missing product price observations (no successful sales for a specific product on a given day) have their values computed using a radial basis function interpolation. No consideration is made for variations in lead-time among product price observations.
4. *Daily price observations for components* are the prices of each of the 16 components, as observed by the agent. The prices are added to the feature vector for a pre-defined schedule of lead times of 2, 6, 10, 20, and 30 days. Days with missing component observations on these lead times will have their values computed using a radial basis function interpolation over observations for the component from the previous 5 days.

These input features combine to produce a vector of 101 features (see Table 1). Labeled training instances are generated by computing the matrix of observations for the features

and appending the known true value (label) for the regression target. This labeled training set can be read by the PLS algorithm to produce a regression model.

Class	Lagged Offsets					
	0	1	2	4	8	16
P3	•					
P3b	•					
P3c	•					
P2	•					
P4	•					
P4h	•					

Table 2: A simple configuration for product prediction. A “•” denotes the feature class for the specified time lag is included.

3.3 Lagged Features and Hierarchical Segmentation

Using only the most recent values of the 101 possible features as the entire feature set may provide reasonable prediction results in some domains, but, it cannot predict trends or temporal relationships present in the data. The need to represent temporally-offset relationships motivates the idea of adding time-delayed observations to the feature set as well; we refer to this as the addition of *lagged features*. For instance, if it is known that the cost of a component on day $t - 8$ is most representative of the price of a product sold on day $t + 1$, the 8 day delayed observation of that component should have a high weight in the model. The time delay could correspond to the delay between persistent changes in the observations and the resulting effect in the mean agent behavior.

Our technique uses the assumption that more recent observations are likely to have high informational value for price prediction, but time-delayed features may hold informational value as well (i.e. the environment is not completely stochastic). The lead time between a change in the market and its effect on other prices may be longer than one day, but the minimum possible lead time for this effect is 1 day but could vary due to the design of opponent agents. To cover the possibility of changes in agent design outside of our control, we keep most recent (current day) observations in the model. For reasons of tractability, features with long time delays are only added after features with shorter delays are added. Even with this constraint, searching for the optimal subset from 101 available features is still an intractably large search space.

To reduce the number of possible configurations of features, we introduce the notion of a hierarchical segmentation of the feature set. In this domain, each of the 101 features is placed into one of several classes based on its relationship to the target variable. In cases where a feature belongs to multiple classes, the feature is placed in the class that is most specific. From a minimal amount of domain knowledge derived from the specification document of TAC SCM, we have compiled a class hierarchy for each type of target variable. These hierarchies can be seen for the product prediction and component prediction tasks in Figures 2 and 3, respectively.

Thus far we have explained how a better feature set can be chosen based on some knowledge of the domain. Now we will

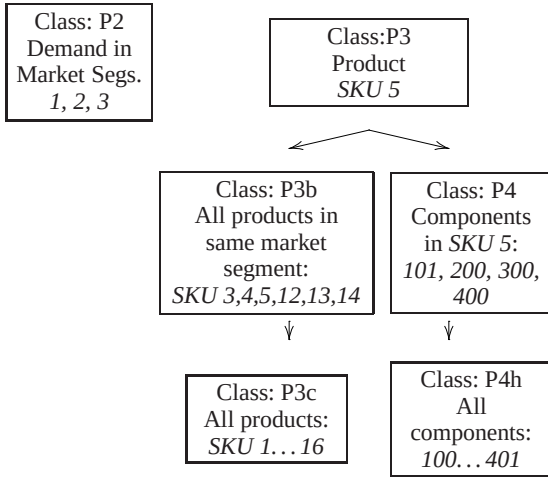


Figure 2: Lag scheme class hierarchy for product price prediction. Arrow denotes a *not greater than* relationship (i.e. class P4 should have an equal or lower maximum time offset than class P3).

address how to choose the time-delayed data from each class. The simplest lag configuration, shown in Table 2, contains the most recent day’s value from all feature classes. We posit that time-delayed observations from the variable of interest (*Class P3*) are likely to be predictive as well. Time-delayed observations from other feature classes *may also be* but are less likely to be predictive. It is by this principle that the hierarchy and strict ordering of lagged data additions is based.

Also, it is necessary to choose the set of permitted lagged offsets from each class. In this domain, two sets of time-delays were found to be useful: time-delays with integer offsets $\{\emptyset^3, 0, 1, 2, 3, 4, 5, \dots\}$ and time-delays with geometrically increasing offsets $\{\emptyset, 0, 1, 2, 4, 8, 16, \dots\}$.

We continue with a discussion of the relationships formulated in the class hierarchy for a specific product in this domain, SKU 5. In Figure 2, the earlier observations of the variable to be predicted (*Class P3*) are most likely to contain predictive information. Information about other similar products (*Class P4*) will also provide some information (but likely are less informationally dense). Finally, information about all other products is expected to contain the least information density (*Class P4h*). By constraining the classes so that the less informationally dense classes have lower time delays and contribute fewer additional features, we prevent the inclusion of extraneous, irrelevant features.

Next, we will show how the time lagged data is constructed to form the augmented features set. An expansion of the feature set is referred to as a *lag scheme expansion*. Of course, the optimal lag scheme may be different for each variable modeled; a search of the possible configurations is performed to find the best performing configuration in each prediction class.

The number of possible lag schemes as formulated with the hierarchies in Figures 2 and 3 for a maximum time delay of 16 days are 11172 for both products and components. With-

³This symbol refers to the lack of any observation.

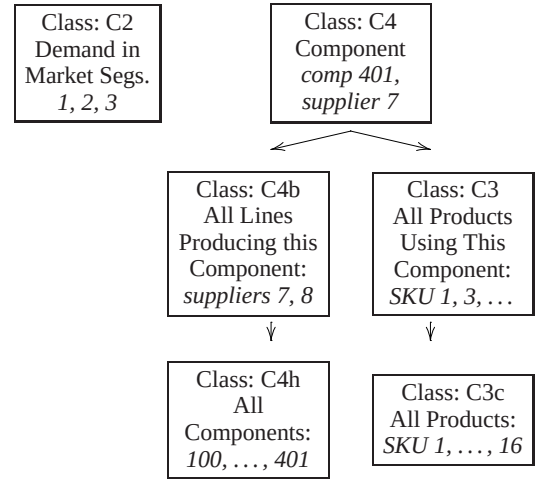


Figure 3: Lag scheme class hierarchy for component prediction.

out the constraints between classes, there are 114649 configurations of the 6 feature classes if constrained to possible time delays of $\{\emptyset, 0, 1, 2, 4, 8, 16\}$, but many of these configurations will be uninteresting variants. Finally, without the hierarchical segmentation and constraints between classes, there are $7^{101} (\approx 10^{85})$ configurations of the 101 original features. This imposing both the feature classification and the constraint hierarchy allows for a greater range of “interesting” lag schemes to be tested for the same amount of lag scheme search.

The optimal lag schemes we use in our experimental results in each of the four prediction classes is provided in Table 3⁴. While a domain expert could conceive of a generally high-performance feature set, the automated lag scheme search should contain a configurations similar to what a domain expert can build (without the cost of becoming a domain expert). Also, the results of the optimal lag scheme search can elicit some surprising relationships found in the data.

3.4 Partial Least Squares Regression

The methodology presented here does not make alterations to the PLS algorithm, so our treatment of PLS will be brief. Several implementations of PLS exist [de Jong, 1993; Dayal and MacGregor, 1997; Martens and Næs, 1992]; each with its own performance characteristics. This work uses the orthogonalized PLS, Non-Integer Partial Least Squares (NIPALS), implementation presented in [Wold, Martens, and Wold, 1983]. PLS was chosen over similar multivariate techniques including multiple linear regression, ridge regression [Hoerl and Kennard, 2000], and principal component regression [Jolliffe, 1982] because it produces generally equiv-

⁴Current product prediction (Table 3(a)) uses 37 raw features that are expanded into 57 augmented features. Feature classes contributing features: P3 (1 raw feature * 10 discrete lags), P3b (5 raw features * 1 discrete lag), P3c (10 raw features * 1 discrete lag), P2 (3 raw features * 5 discrete lags), P4 (4 raw features * 1 discrete lag), P4h (11 raw features * 1 discrete lag), and game invariant features (game storage cost, game interest rate).

Algorithm 1: PLS1 Calibration

input : A matrix X containing n training samples, n rows each with m features. The corresponding vector y containing n labels for the training set. Model complexity A_{max} chosen so that $a = 1, \dots, A_{max}$.

output: Loading arrays $\hat{W}$, $\hat{Q}$, and $\hat{P}$.

STEP 1 $X_0 = X - 1\bar{x}'$, where $\bar{x}'$ is a vector of the mean values of the variables in X .
 $y_0 = y - 1\bar{y}$, where $\bar{y}$ is the mean value of y

optional Normalize columns in X_0 to have equivalent variance. (Divide each column by its variance.)

for $a = 1 \rightarrow A_{max}$ **do**

STEP 2.1 Using least squares, compute normalized local model $\hat{w}_a$
 $\hat{w}_a = X'_{a-1}y_{a-1} / \|X'_{a-1}y_{a-1}\|$

STEP 2.2 Estimate scores $\hat{t}_a$ using model $\hat{w}_a$.
 $\hat{t}_a = X_{a-1}\hat{w}_a$ (since $\hat{w}_a'\hat{w}_a = 1$)

STEP 2.3 Estimate x -loadings $\hat{p}_a$ using scores $\hat{t}_a$.
 $\hat{p}_a = X'_{a-1}\hat{t}_a / \hat{t}_a'\hat{t}_a$

STEP 2.4 Estimate y -loadings $\hat{q}_a$ using scores $\hat{t}_a$.
 $y_{a-1} = \hat{t}_a\hat{q}_a + f$
 $\hat{q}_a = y'_{a-1}\hat{t}_a / \hat{t}_a'\hat{t}_a$

STEP 2.5 Update X and y with contribution of current a .
 $X_a = X_{a-1} - \hat{t}_a\hat{p}_a'$
 $y_a = y_{a-1} - \hat{t}_a\hat{q}_a$

end

Algorithm 2: PLS1 Prediction

input : Populated feature vector x_i , calibration $\bar{x}$, calibration $\bar{y}$, loading weights $\hat{W}$, loadings $\hat{P}$, loadings $\hat{Q}$

output: Prediction of $\hat{y}_i$

STEP 1 Center observation of feature vector x_i .
 $x_{i,0} = x_i - 1\bar{x}'$

optional Normalize variance $x'_{i,0}$ by dividing each value by its column variance in calibration X .

for $a = 1 \rightarrow A_{max}$ **do**

STEP 2.1 Compute contribution of $\hat{w}_a$ to y_i .
 $\hat{t}_{i,a} = x'_{i,a} - 1\hat{w}_a$
 $x_{i,a} = x_{i,a-1} - \hat{t}_{i,a}\hat{p}_a'$

end

STEP 3 Compute prediction of y_i
 $\hat{y}_i = \bar{y} + \sum_{a=1}^A \hat{t}_{i,a}\hat{q}_a$

Alt. Alternative formulation in 1 step using $\hat{b}$.
 $\hat{b}_0 = \bar{y} - \bar{x}'\hat{b}$
 $\hat{B} = \hat{W}(\hat{P}'\hat{W})^{-1}\hat{Q}$
 $\hat{y}_i = 1\hat{b}_0 + x'_i\hat{B}$

Class	Lagged Offsets									
	0	1	2	3	4	5	6	7	8	9
P3	•	•	•	•	•	•	•	•	•	•
P3b	•									
P3c	•									
P2	•	•	•	•	•					
P4	•									
P4h	•									

(a)

Class	Lagged Offsets					
	0	1	2	4	8	16
P3	•	•	•	•	•	
P3b	•	•	•	•	•	
P3c						
P2	•	•	•	•	•	
P4	•					
P4h						

(b)

Class	Lagged Offsets									
	0	1	2	3	4	5	6	7	8	9
C4	•	•	•	•	•	•	•	•		
C4b										
C4h										
C3										
C3c										
C2										

(c)

Class	Lagged Offsets					
	0	1	2	4	8	16
C4	•	•	•	•	•	
C4b						
C4h						
C3	•	•	•	•	•	
C3c						
C2	•	•	•	•	•	

(d)

Table 3: Shows the optimal lag schemes for the the four prediction classes: (a) current product prediction (expands to 57 augmented features), (b) future product prediction (51 augmented features), (c) current component prediction (10 augmented features), and (d) future component prediction (102 augmented features).

alent or better performance than the others and has the ability to adjust model complexity. Specific advantages of this algorithm are presented.

Partial least squares regression is particularly applicable to modeling economic phenomena. First, PLS regression is able to handle very high-dimensionality inputs because it implicitly performs dimensionality reduction from the number of inputs to the number of PLS factors. Second, the model complexity can be adjusted by changing the number of PLS factors to use in computing the regression result. This value is adjusted in our experiments to determine the optimal model complexity in each prediction class. Third, the algorithm is generally robust to highly collinear or irrelevant features.

Mathematically, PLS regression deterministically computes a linear function that maps a vector of the input features x_i into the output variable y_i (the label). Using a PLS regression model for a particular variable y requires a calibration to

	Current Product	Future Product (p -val)	Current Component	Future Component (p -val)
PLS (geom. lag) RMSE	0.06308	0.08553 (0.2660)	0.04063	0.09910 (0.0012)
PLS (int. lags) RMSE	0.06295	0.09264	0.04008	0.10020
PLS (no lag) RMSE	0.06300	0.09039	0.04468	0.10947
2008 First Place RMSE	0.04779	0.08726	0.03476	0.09964
2008 Third Place RMSE	0.0531	0.09934	0.04029	0.10281

Table 4: Comparison between the error scores of an agent implementing our method for each of the 4 classes of prediction and the scores of the top performing agents in the 2008 Prediction Challenge. Values in parenthesis indicate Student’s t -test p -value when compared against 2008 First Place agent. A t -test is only provided for our algorithm when it performs better than the 2008 published results. A p -value (p -val) of less than 0.05 indicates that the results of the two algorithms being compared are statistically different at the 95% confidence level.

be performed over a set of training samples to determine the model as computed using Algorithm 1. The model can be used for prediction as described mathematically in Algorithm 2.

4 Model parameter: Dimensionality Reduction

The PLS regression algorithm in [Wold, Martens, and Wold, 1983] allows users to adjust the model complexity by selecting the number of PLS factors to generate when training. (These factors are analogous to the principal component vectors used in principal component regression.) The number of PLS factors determines the dimensionality of the intermediate variable space that the data is mapped to. The computational complexity does not significantly increase for a larger number of factors but the choice does have an effect on prediction performance: too large a number can cause over-fitting, and too small a number can cause the model to be unable to represent the relationships in the data. The number of factors was systematically varied in the optimal lag scheme search. The best performing number of components is shown for each prediction category in Section 5.

When computing the optimal lag scheme it is also critical to determine the correct value for model complexity in PLS. Figure 4 shows how the prediction accuracy varies based on model complexity relative to the best observed prediction error (the lag scheme is not varied in the data in this graph). The future component class has a slightly different pattern: we conjecture that future component achieves optimal error with a lower number of latent variables because it has a relatively larger number of inputs from the lag scheme search. Empirical results show that it is generally better to err on the side of excess model complexity but excess complexity can also reduce prediction accuracy by, in one case, over 10% above the lowest achievable error.

5 Experimental Results

We evaluate our approach on the TAC SCM 2008 Prediction challenge data set consisting of 48 experimental runs divided into 3 sets of 16 games. Each set has a different mixture of agents. The games were divided using a standard 6-fold cross validation for training and scoring. For example, the first fold consisted of a training set of games 9-16 in set A and all games from sets B and C, and a test set of games 1-8 in set A.

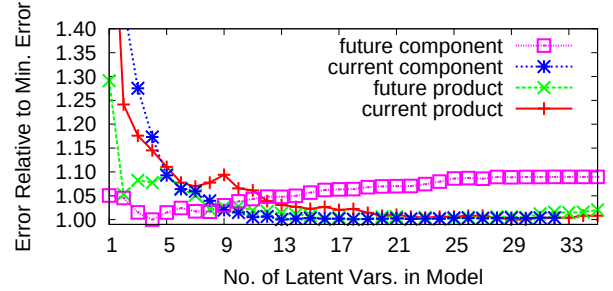


Figure 4: Effect of varying the model complexity (number of PLS factors) for the optimal lag scheme. The number of factors used in our experiments varied by prediction class: (1) current product used 18 factors, (2) future product used 22 factors, (3) current component used 22 factors, and (4) future component used 4 factors.

In order to compute prices for all variables in each prediction class, one prediction model is generated for each variable to predict. The number of prediction models by class is:

1. 16 for current product prices (one for each product type),
2. 16 for future product prices,
3. 16*59 for current component prices (one for each component line-lead time pair for lead times 2 to 60), and
4. 16*26 for future component prices (lead times 5 to 30)

The lag scheme used for each prediction class was computed using an exhaustive search over all lag schemes with valid hierarchical relationships on folds 1, 3, and 5 of the 6-fold cross validation set. The results shown in Table 4 are the aggregate scores obtained from experiments on the entire 6-fold set using the best performing lag schemes. The results show that incorporating time-delayed features onto the prediction task does improve the overall prediction accuracy. A comparison with the published results shows that our method is competitive against state-of-the-art methods used within the top performing agents. In particular, our method outperforms existing methods on 20 day ahead predictions in both the component market and the product market. The prediction category for which our method does worst (“current product”) can be explained by the fact that the regression model computes a point estimate for sales of each product on each day.

There may be some attribute in product requests that causes prices to vary significantly among the set of requests for the same product on the same day. For “future product”, a point estimate is sufficient because the true value represents the median price for product sales 20 days ahead.

The optimal lag schemes used in each of the four prediction classes is provided in Table 3. There are some feature classes that are not included at all in the optimal lag scheme, an example of this appears in Table 3(c): lagged data from class C4 are the only features used in the prediction model. This suggests that the other variables are not significant to the prediction of current component prices.

6 Conclusions and Future Work

Prediction problems that are dynamic and possess high dimensionality are generally difficult to model well. It is easy to disregard predictive information sources in exchange for model simplicity. Imposing a class-based feature hierarchy and using time-delayed data extraction in conjunction with a multivariate regression approach provides a modeling framework that maximizes the use of available predictive inputs while being robust to irrelevant data. Hierarchical classification of features is of benefit because it reduces the search space for finding the best lag configurations.

We believe that the time-delayed data extraction and classification is not tied to PLS Regression for good performance. But, PLS regression was chosen because of its desirable properties for high-dimensional domains. In applications where performance is a significant consideration, it may be possible to compute the regression results for several target variables simultaneously in a single model PLS model using a variant called PLS2 regression. Also, the mathematical description of the trained regression model itself can be leveraged to develop domain knowledge about relationships between variables in the domain. More opaque, “black box” machine learning models may not facilitate this. Additionally, this information could be used for manual validation of the computed models. Also, this analysis may facilitate better decision making in the application domain. In TAC SCM, empirically constructed causal models may allow agent developers to focus on specific behaviors that affect prices.

We believe this approach is highly applicable to other economic data sets. In particular, this data preparation and prediction method may have application in cases where there are significant spatial relationships between input sources. Examples of such domains include airline ticket pricing and river-flow prediction. This is an area left for future work.

References

- [Benisch et al., 2009] Benisch, M.; Sardinha, A.; Andrews, J.; Ravichandran, R.; and Sadeh, N. 2009. Cmieux: Adaptive strategies for competitive supply chain trading. *Electronic Commerce Research and Applications* 8(2):78–90.
- [Box, Pelham, and Jenkins, 1994] Box, G.; Pelham, E.; and Jenkins, G. M. 1994. *Time Series Analysis: Forecasting and Control*. Prentice Hall PTR, 3rd edition.
- [Collins et al., 2006] Collins, J.; Arunachalam, R.; Sadeh, N.; Ericsson, J.; Finne, N.; and Janson, S. 2006. The supply chain management game for the 2006 trading agent competition. Technical Report CMU-ISRI-07-100, Carnegie Mellon University, Pittsburgh, PA.
- [Dayal and MacGregor, 1997] Dayal, B. S., and MacGregor, J. F. 1997. Recursive exponentially weighted pls and its applications to adaptive control and prediction. *Journal of Process Control* 7(3):169–179.
- [de Jong, 1993] de Jong, S. 1993. Simpls: An alternative approach to partial least squares regression. *Chemometrics and Intelligent Laboratory Systems* 18(3):251–263.
- [Hillmer, Larcker, and Schroeder, 1983] Hillmer, S. C.; Larcker, D. F.; and Schroeder, D. A. 1983. Forecasting accounting data: A multiple time-series analysis. *Journal of Forecasting* 2(4):389–404.
- [Hoerl and Kennard, 2000] Hoerl, A. E., and Kennard, R. W. 2000. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics* 42(1):pp. 80–86.
- [Jolliffe, 1982] Jolliffe, I. T. 1982. A note on the use of principal components in regression. *J. of Royal Statistical Society. (Applied Statistics)* 31(3):pp. 300–303.
- [Kearns and Ortiz, 2003] Kearns, M., and Ortiz, L. 2003. The Penn-Lehman Automated Trading Project. *IEEE Intelligent Systems* 18(6):22–31.
- [Ketter et al., 2009] Ketter, W.; Collins, J.; Gini, M.; Gupta, A.; and Schrater, P. 2009. Detecting and forecasting economic regimes in multi-agent automated exchanges. *Decision Support Systems* 47(4):307–318.
- [Kiekintveld et al., 2009] Kiekintveld, C.; Miller, J.; Jordan, P. R.; Callender, L. F.; and Wellman, M. P. 2009. Forecasting market prices in a supply chain game. *Electronic Commerce Research and Applications* 8(2):63–77.
- [Martens and Næs, 1992] Martens, H., and Næs, T. 1992. *Multivariate Calibration*. John Wiley & Sons.
- [Olason and Watt, 1986] Olason, T., and Watt, W. E. 1986. Multivariate transfer function-noise model of river flow for hydropower operation. *Nordic Hydrology* 17(3):185–202.
- [Pardoe and Stone, 2007] Pardoe, D., and Stone, P. 2007. An autonomous agent for supply chain management. In Adomavicius, G., and Gupta, A., eds., *Handbooks in Information Systems Series: Business Computing*. Elsevier.
- [Pardoe and Stone, 2008] Pardoe, D., and Stone, P. 2008. The 2007 TAC SCM prediction challenge. In *AAAI 2008 Workshop on Trading Agent Design and Analysis*.
- [Stone and Greenwald, 2005] Stone, P., and Greenwald, A. R. 2005. The first international trading agent competition: Autonomous bidding agents. *Electronic Commerce Research* 5(2):229–265.
- [Tiao and Box, 1981] Tiao, G. C., and Box, G. E. P. 1981. Modeling multiple time series with applications. *Journal of the American Statistical Association* 76(376):802–816.
- [Wold, Martens, and Wold, 1983] Wold, S.; Martens, H.; and Wold, H. 1983. The multivariate calibration problem in chemistry solved by the PLS method. In *Matrix Pencils*, volume 973 of *LNM*. Springer. chapter 18, 286–293.

Testing Adaptive Expectations Models of a Continuous Double Auction Market Against Empirical Facts

Neil Rayner ⁽¹⁾, Steve Phelps ⁽¹⁾ and Nick Constantinou ⁽²⁾

⁽¹⁾ Center for Computational Finance and Economic Agents (CCFEA)

⁽²⁾ Essex Business School (EBS)

University of Essex, Colchester, CO4 3SQ, United Kingdom

Email: njwray@essex.ac.uk

Abstract

It is well known that empirical financial time series data exhibit long memory phenomena: the behaviour of the market at various times in the past continues to exert an influence in the present. One explanation for these phenomena is that they result from a process of social learning in which poorly performing agents switch their strategy to that of other agents who appear to be more successful. We test this explanation using an agent-based model and we find that the stability of the model is directly related to the dynamics of the learning process; models in which learning converges to a stationary steady state fail to produce realistic time series data. In contrast, models in which learning leads to *dynamic* switching behaviour in the steady state are able to reproduce the long memory phenomena. We demonstrate that a model which incorporates contrarian trading strategies results in more dynamic behaviour in steady state, and hence is able to produce more realistic results.

1 Introduction

With the explosion of algorithmic trading, financial markets now constitute some of the largest and most mission critical multi-agent systems in our society. Understanding the behaviour of these markets would make an important contribution to the prevention of future financial crises. There is a need to build models of *actual* electronic markets, and to validate these models against empirical facts – that is, to attempt to *reverse engineer* existing multi-agent systems in order to understand how they work. Such an exercise is now possible with the availability of electronic data detailing every transaction in the market which can run to gigabytes per year per financial asset, and can be purchased from the major financial exchanges by any third party. Towards this end we introduce an agent-based model which produces behaviour consistent with several phenomena that have been widely documented from studies of empirical financial data. Our model is in the tradition of *adaptive expectations* [Evans and Honkapohja, 2001] models in which: (i) agents' valuations are determined by their expectations of what will happen in the market in the *future*, for example their belief that the market price will rise

or fall; and (ii) expectations are formed inductively through a learning process, rather than through the framework of rational expectations. This type of model is in contrast to auction theoretic models which typically assume that valuations are private information, are well defined, uncorrelated, or do not change over time, or some combination of these. In contrast the adaptive expectations framework proposes a much more dynamic view of agents' beliefs as they constantly revise their expectations, and hence valuations, in response to observations of other agents and the market itself: the market is an "expectations feedback system" from which valuations emerge [Heemeijer *et al.*, 2004].

The focus of our analysis is to determine to what extent this picture of the market is consistent with the empirical data from real exchanges. Initially we examine properties which are observable in empirical high frequency trading data with a view to model validation. Once our model is validated, we can use it to answer counterfactual questions such as how changes in the design of the market mechanism would affect the efficiency of the market. Therefore we focus on well known "stylized facts" of high-frequency time series data observed in real financial markets and we analyse to what extent different model assumptions are consistent with these phenomena. Specifically we analyze long memory in the following attributes:

1. *Volume*: Over periods of time volume can be consistently high or low [Lobato and Velasco, 2000];
2. *Volatility*: Periods of similar volatility (price fluctuations) are observed [Ding *et al.*, 1993; Lamer, 1990; Mantegna and Stanley, 2002; Pagan, 1996];
3. *Order Signs*: Signs of orders (that is, buy orders have a positive sign and sell orders have a negative sign) exhibit long memory [Lillo *et al.*, 2005]; and
4. *Returns*: Returns do not exhibit long memory [Cont, 2001]. (Similar returns do not cluster together in time).

In Section 2 we describe an existing agent-based model [LeBaron and Yamamoto, 2007; 2008] within which agents learn trading strategies from each other. These landmark experiments demonstrated the power of agent-based modelling and specifically its ability to address poorly understood phenomena like long memory in high frequency financial data. We extend this model by introducing more contrarian fore-

casting strategies and demonstrate that this leads to an improvement in the robustness of the model. We validate the existing LeBaron and Yamamoto model in Section 3 and present our results in Section 4. In Section 5 we compare the ability of the LeBaron and Yamamoto model and our more contrarian model to produce long memory “stylized facts” of financial markets. Finally in Section 6 we conclude.

2 The Model

We attempt to explain the long memory phenomena using an adaptive expectations model with three classes of strategy which are used to form expectations about future returns:

1. *fundamentalists* value a stock through an understanding of its hypothetical underlying value, in other words, based on expectations of the long term profitability of the issuing company;
2. *chartists* form valuations inductively from historical price data; and
3. *noise traders* who trade based on the fluctuations of the price of a stock. The buying and selling behaviour of traders for a particular stock generates characteristic fluctuations in price. A stock has an emergent volatility, understanding this volatility allows traders to identify when the price is relatively low and when it is relatively high.

Although chartist strategies should not be profitable according to the efficient markets hypothesis, this is not necessarily true if the market is outside of an efficient equilibrium. For example, if many agents adopt a chartist forecasting strategy it may be rational to follow suit as the chartist expectations may lead to a self-fulfilling prophecy in the form of a speculative bubble. Thus there are feedback effects from these three classes of forecasting strategy and it is important to study the interaction between them in order to understand the macroscopic behaviour of the market as a whole.

We model the market mechanism as a continuous double auction with limit orders. Each agent submits a limit order to the market on every round of trading. Orders are executed using a time priority rule: the transaction price is the price of the order which was submitted first regardless of whether it is a bid or ask. If an order cannot be executed immediately it is queued on the order-book.

The sign (buy or sell) and the price of the order for agent i at time t is determined as a function of each agent’s *forecast* of expected return $\hat{r}_{(i,t,t+\tau)}$ for the period $t + \tau$ (τ a constant defining the time horizon over which price expectations are made). The price of the order is set according to:

$$p_{(i,t+\tau)} = p_t \cdot e^{\hat{r}_{(i,t,t+\tau)}}$$

where p_t is the market quoted price at time t , and the sign of the order is buy iff. $p_{(i,t+\tau)} \geq p_t$ or sell iff. $p_{(i,t+\tau)} < p_t$.

In [LeBaron and Yamamoto, 2007; 2008] the forecasted expected return for the period $t + \tau$ of agent i at time t is calculated with a linear combination of fundamentalist, chartist and noise-trader forecasting rules:

$$\hat{r}_{(i,t,t+\tau)} = \hat{r}_{f(i,t,t+\tau)} + \hat{r}_{c(i,t,t+\tau)} + \hat{r}_{n(i,t,t+\tau)} \quad (1)$$

$$\hat{r}_{f(i,t,t+\tau)} = f_{(i,t)} \cdot \left(\frac{F - p_t}{p_t} \right) \quad (2)$$

$$\hat{r}_{c(i,t,t+\tau)} = c_{(i,t)} \cdot \hat{r}_{L_i} \quad (3)$$

$$\hat{r}_{n(i,t,t+\tau)} = n_{(i,t)} \cdot \epsilon_{(i,t)} \quad (4)$$

where F is the so-called “fundamental price” (which is exogenous and fixed for all agents), p_t is the current market quoted price which is the value of the transaction at the previous time step or in the absence of a transaction the mid-point of the spread, $\epsilon_{(i,t)}$ are random iid. variables distributed $\sim N(0, 1)$ and r_{L_i} is a forecast based on historical data; in our case a moving average of actual market returns over the period L_i :

$$\hat{r}_{L_i} = \frac{1}{L_i} \sum_{j=1}^{L_i} \frac{p_{t-j} - p_{t-j-1}}{p_{t-j-1}}$$

The period L_i is randomly and uniformly initialised from the interval $(1, l_{max})$. The linear coefficients $f_{(i,t)}$, $c_{(i,t)}$ and $n_{(i,t)}$ denote the weight that agent i gives to each class of forecast amongst fundamentalist, chartist and noise-trader respectively at time t . Bids (b_t^i) and asks (a_t^i) (that is buys and sells) are entered into the with a markup or markdown.

$$b_t^i = p_{t,t+\tau}^i (1 - k^i) \quad (5)$$

$$a_t^i = p_{t,t+\tau}^i (1 + k^i) \quad (6)$$

k^i is randomly and uniformly initialised from an interval $(0, k_{max})$.

1. If a bid exceeds the best ask (lowest ask price on the order book) it is entered at the ask price (converted into a market order rather than a limit order).
2. If an ask is lower than the best bid (highest bid price on the order book) it is entered at the bid price (converted into a market order rather than a limit order).

Each agent enters the market with a constant probability λ . All orders have a limited order life, after which they are removed from the order book should they have not been successfully matched (a constant exogenously set to 200 units of time).

2.1 Learning

As in [LeBaron and Yamamoto, 2007; 2008], agents use a co-evolutionary Genetic Algorithm to learn the coefficients $f_{(i,t)}$, $c_{(i,t)}$ and $n_{(i,t)}$. Each agent records its own forecast error as the market progresses and generates a fitness score s_i over a period of 5000 units of time. Each unit of time corresponds to the entry of an order into the market by an agent. Each agent presents 5 orders to the market in this period (the number of agents in these models being 1000).

$$s_i = \frac{1}{\sum_{t=1}^{5000} (p_t - p_{(i,t+\tau)})^2} \quad (7)$$

After 5000 units of time each combination of weights held by the agents is assigned a relative fitness score (S_i) normalised with respect to the population fitness.

$$S_i = \frac{s_i}{\sum_i s_i} \quad (8)$$

The strategy weights are copied by the learning agents in proportion to this score.

The *initial* values at time $t = 0$ for the fundamentalist $f_{(i,0)}$, chartist $c_{(i,0)}$ and noise $n_{(i,0)}$ weights are drawn from the following distributions:

$$f_{(i,0)} \sim |N(0, \sigma_f)|, \quad c_{(i,0)} \sim N(0, \sigma_c), \quad n_{(i,0)} \sim |N(0, \sigma_n)| \quad (9)$$

In addition to the learning of weights after each 5000 units of time, agents also may mutate one of their weights drawing a weight at random from the distributions in Equation 9.

We analyse two variants of this basic model; an existing model in the literature [LeBaron and Yamamoto, 2007; 2008] in which forecasting strategies are linear *combinations* as per Equation 1 (henceforth we refer to this model as the LY model), and our own model in which each agent adopts either an *atomic* fundamentalist (Equation 2), chartist (Equation 3) or noise trader (Equation 4) forecasting rule and not a linear combination as in the LY model above (Equation 1). Both our model and the LY model occupy the same strategy space. However, in our model, two out of the three weights are zero reducing each agent to just one of the return forecast rules (Equation 2, Equation 3 or Equation 4).

In [Conrad, 1998] two main types of strategy are identified; momentum strategies based on the following of trends, and contrarian strategies based on the reversal of trends. Contrarian traders predict price reversals and make profit (when they are correct) by positioning themselves to take advantage of that reversal. These two diametrically opposed strategies appear to exist simultaneously [Conrad, 1998]. For example, in a rising trend momentum strategists will place bid orders (buys) in the market while contrarian strategists will place asks orders (sells). Other factors contribute to contrarian like behaviour. For example, “pairs trading” where a pair of related stocks are traded together such that when one is relatively expensive and the other cheap, traders sell the expensive one and buy the cheap one which can be a behaviour which is entirely independent of the current trends in the market. Other contrarian-like behaviours can be caused by events external to the market, for example arbitrage opportunities; prices for the same stock can differ between two markets so traders buy in the cheaper market and sell in the more expensive market (gaining risk-free profit). This arbitrage behaviour can be counter to the trends in the market.

The LY Model implements contrarianism by allowing the chartist weight to go negative (Equation 9). In the LY Model agents imitate successful strategies: if the most successful strategy employs a negative chartist weight then agents will tend to adopt that strategy. The negative chartist weight

would then represent not a contrarian position but the strategy of the momentum strategists. Contrarian strategists seek to behave in a contrary way to the momentum strategists. To capture this behaviour we introduce additional contrarianism. We add two contrarian strategies; one is to negate the learned trend, so the contrarian agent predicts a price move in the opposite direction of the learned trend (e.g if the learned chartist trend is negative then the contrarians will predict a price move in a positive direction and vice versa), and the second to zero the trend predicting that the price will not trend in the learnt direction at all but will remain at its current level. We have chosen to implement this firstly by setting the contrarian chartist strategy to the negative of the non-contrarian chartist strategy:

$$\hat{r}_{c_c(i,t,t+\tau)} = -\hat{r}_{c(i,t,t+\tau)}$$

Secondly we set the contrarian fundamentalist and noise strategies to be the zeroed non-contrarian fundamentalist and noise strategies:

$$\hat{r}_{f_c(i,t,t+\tau)} = \hat{r}_{n_c(i,t,t+\tau)} = 0$$

In the *contrarian* variant agents can choose from the following discrete set of return forecasting strategies:

$$\{\hat{r}_{c(i,t,t+\tau)}, \hat{r}_{f(i,t,t+\tau)}, \hat{r}_{n(i,t,t+\tau)}, \hat{r}_{f_c(i,t,t+\tau)}, \hat{r}_{n_c(i,t,t+\tau)}, \hat{r}_{c_c(i,t,t+\tau)}\}$$

The same learning process operates in this model as in the LY model (but with the addition of the contrarian parameter). So an agent can change from fundamentalist to chartist or contrarian to non-contrarian to take advantage of a better strategy. During initialisation of the model values are drawn randomly from the distributions in Equation 9 as in the LY model, but each agent also chooses randomly between being a fundamentalist, chartist or noise trader and contrarian or non-contrarian. Henceforth we refer to this latter model as “the Contrarian Model”.

3 Methodology and Model validation

We compare model assumptions according to how well a particular model reproduces the long-memory phenomena. To compare models we test their long-memory properties using Lo’s modified rescaled range (R/S) statistic [Lo, 1991] (sometimes called range over standard deviation). The statistic is designed to compare the maximum and minimum values of running sums of deviations from the sample mean, re-normalized by the sample standard deviation. The deviations are greater in the presence of long-memory than in the absence of long-memory. The Lo R/S statistic includes the weighted auto-covariance up to lag q to capture the effects of short-range dependence.

Our first experiment tests the conjecture that imitation (in the LY Model) is sufficient to produce the long memory phenomena by attempting to reject the null hypothesis that long memory is caused by the strategies that each agent adopts and

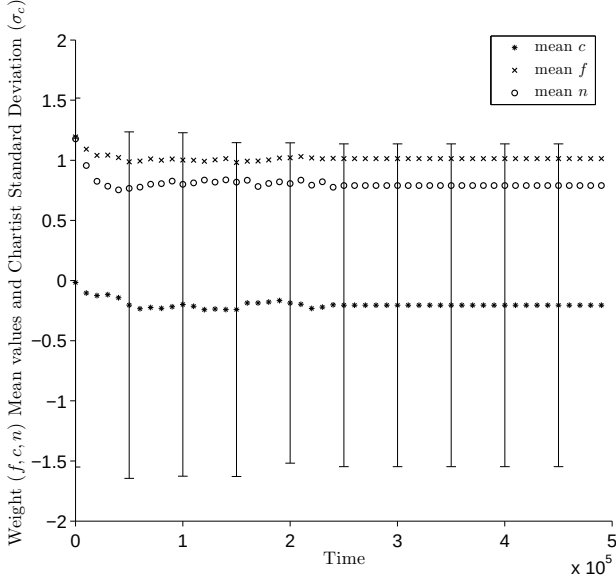


Figure 1: LY Model. Fundamentalist (f), Chartist (c), Noise (n) mean weights and Chartist Standard Deviation (σ_c) with time.

not by learning at all. The model is simulated in two sequential phases with different treatment factors:

1. a learning phase in which the agent's genetic algorithm searches for strategies with high relative fitness (see Equation 7 and Equation 8).
2. a commitment phase where agent's commit to a learned strategy and perform no further learning.

The default experiment time is 250000 time units (taken from [LeBaron and Yamamoto, 2007; 2008]). The experiment has been executed for twice the default experiment time (2 x 250000 units of time); the learning phase is executed for half the experiment time (the default time) and then the commitment phase is started and runs for the same period (default parameter values are displayed in Table 1 these values have been taken from [LeBaron and Yamamoto, 2007; 2008] except the standard deviations of the weight distributions which are all set to 1.5 and the probability of entering the market which is set to 1.0. These values are all varied in our tests on model stability).

4 Validation Results for LY Model

In Figure 1 we show the mean value (across all agents) of the fundamentalist (f), chartist (c) and the noise trader weight (n) with respect to time. It also shows the chartist weight distribution standard deviation (σ_c) (the noise and fundamentalist weight distribution standard deviation are not shown but behave in a similar manner). As we can see in Figure 1 the agents move initially very quickly to a region in the strategy space. There is then a period of mean fluctuation as the agents move about in that region (not converging to any specific strategy). When the commitment phase starts and the

Table 1: Default Values for All Models.

Parameter	Value
Std dev of fundamental weight (σ_f)	1.5
Std dev of chartist weight (σ_c)	1.5
Std dev of noise weight (σ_n)	1.5
Probability of Mutating	0.08
Probability of entering the λ	1.0
Maximum markup or markdown (k_{max})	50%
Maximum period over which trends are calculated (l_{max})	100 units of time
Period over which price expectations are made (τ)	200 units of time
Fundamental price (F)	1000
Number of Traders	1000
Order Life	200 units of time
Tick Size (the smallest price differential)	0.1

Table 2: LY Model Learning Phase. Percentages of runs with long memory for volume, volatility, order signs and returns at various time lags ranging From 4×50 (200 units of time) to 10×50 (500 units of time).

Lag	Volume	Volatility	Order Signs	Returns
q=4	100	100	87	4
q=6	100	100	89	4
q=8	100	100	89	4
q=10	100	100	89	4

agents stick with the strategy they have found, movement in the weights cease and we end up with a straight line for the mean value of the weights over the remainder of the experiment time with no change in the weight distribution standard deviations. Results are shown in tables which present the percentage of executions exhibiting long memory in volume, volatility, signs of orders (buy or sell orders) and returns for each experiment. The experiment was executed 100 times; the results are summarised in Tables 2 and 3. In the first phase (the learning phase presented in Table 2) we see the long memory characteristics we are expecting with this model. In the second phase (the commitment phase presented in Table 3) we fail to generate any long memory properties.

As soon as we switch off learning, these long memory phenomena disappear. It is not sufficient to have just the correct mix of strategies in order to generate long memory. So there is something about the dynamics of weight changing (caused in this case by the learning process) which is causing these phenomena.

Table 3: LY Model Commitment. Percentages of runs with long memory for volume, volatility, order signs and returns at various time lags ranging From 4×50 (200 units of time) to 10×50 (500 units of time).

Lag	Volume	Volatility	Order Signs	Returns
q=4	0	0	0	0
q=6	0	0	0	0
q=8	0	0	0	0
q=10	0	0	0	0

Table 4: Ranges of Parameter Values

Parameter	Value
Std dev of fundamental weight (σ_f)	0.0 to 3.0
Std dev of chartist weight (σ_c)	0.0 to 3.0
Std dev of noise weight (σ_n)	0.0 to 3.0
Mutation Constant	0.05 to 0.20

Table 5: Parameter Variation Experimental Results for LY Model. Percentages of runs with long memory for volume, volatility, order signs and returns at various time lags ranging From 4×50 (200 units of time) to 10×50 (500 units of time).

Lag	Volume	Volatility	Order Signs	Returns
q=4	76	86	18	12
q=6	76	83	18	14
q=8	76	82	18	14
q=10	75	81	18	15

5 Model Stability

In this section we review the stability of the LY Model. We vary some of the free-parameters described earlier, the standard deviations of the Gaussian distributions from which the weights are chosen (Equation 9) and the mutation degree. The mutation degree is the probability that any individual will mutate it's strategy and draw a new weight from the distributions in (Equation 9). We have extended the experiment execution time to highlight any problems in stability with respect to time. The experiments were run for 10 times the default time (10×250000 units of time) and parameter values were randomly drawn (uniformly) from the ranges in Table 4. Fifty sets of random parameter variations were executed with 10 executions for each set (totalling 500 for 2500000 units of time).

In Table 5 we present the results of the first experiment. We note we get negative results from the LY Model which produces weak Order Sign long memory but also long memory in returns (not a stylised fact of financial markets).

In Tables 6 and 7 we have separated out the long mem-

Table 6: Early Phase Execution Results for LY Model. Percentages of runs with long memory for volume, volatility, order signs and returns at various time lags ranging From 4×50 (200 units of time) to 10×50 (500 units of time).

Lag	Volume	Volatility	Order Signs	Returns
q=4	100	99	50	50
q=6	100	99	52	55
q=8	100	99	53	57
q=10	100	99	54	61

Table 7: Later Phase Execution Results for LY Model. Percentages of runs with long memory for volume, volatility, order signs and returns at various time lags ranging From 4×50 (200 units of time) to 10×50 (500 units of time).

Lag	Volume	Volatility	Order Signs	Returns
q=4	69	79	27	25
q=6	70	77	28	28
q=8	70	76	29	29
q=10	70	75	29	31

Table 8: Experimental Results for Atomic Model. Percentages of runs with long memory for volume, volatility, order signs and returns at various time lags ranging From 4×50 (200 units of time) to 10×50 (500 units of time).

Lag	Volume	Volatility	Order Signs	Returns
q=4	96	90	43	0
q=6	96	89	41	0
q=8	96	89	40	0
q=10	96	89	39	1

Table 9: Experimental Results for Contrarian Model. Percentages of runs with long memory for volume, volatility, order signs and returns at various time lags ranging From 4×50 (200 units of time) to 10×50 (500 units of time).

Lag	Volume	Volatility	Order Signs	Returns
q=4	94	93	58	0
q=6	94	92	58	0
q=8	93	91	58	0
q=10	92	91	58	0

ory properties of the execution of the LY Model into an early part of the test and a later part. We note that the long memory properties of the model are changing with respect to time. The state of the population in the LY Model converges to a

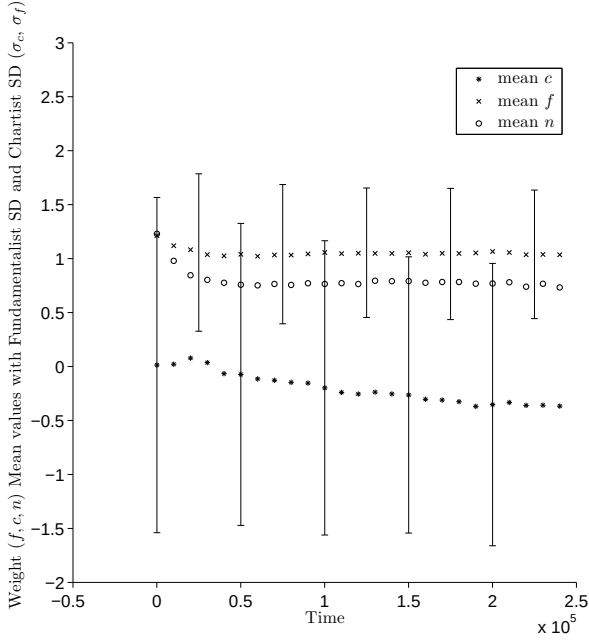


Figure 2: LY Model. Mean Fundamentalist (f), Chartist (c) and Noise (n) weights with time and Fundamentalist and Chartist Standard Deviation (σ_f, σ_c)

relatively static steady-state. At the same time, the long memory properties of the market diminish; the long memory phenomena are not stable over time in the LY Model. In Table 8 we display the results for the execution of the LY Model with just the atomic extensions and finally in Table 9 we display the results for the Contrarian model. Comparing Tables 8 and 9 with the LY Model execution in Table 5 we see a substantial improvement in the stability of the Contrarian model over the LY model.

In Figure 2 the weight means of LY Model change relatively smoothly, the Contrarian Model (Figure 3), in contrast, is very much more dynamic, the mean values for fundamentalist and chartist are moving a great deal relative to the LY model. Comparing the bars (which indicate the fundamentalist SD (σ_f) and chartist SD (σ_c) with time) in Figure 2 and 3 we see that the LY Model fundamentalist and chartist weight distributions tend to converge while with the Contrarian Model the chartist weight distribution hardly converges at all and the fundamentalist distribution is diverging.

We are seeing the convergence of the LY Model into a region in the strategy space. The GA in the LY model has been successful in finding a region in this space (the successful completion of its learning). The Contrarian Model is failing to converge in the strategy space. The LY Model is not generating stable long memory properties, because the successful genetic algorithm is converging to a region in the strategy space. By restructuring the agent strategy space and increasing contrarianism we are able to retain the dynamic necessary to produce stable long memory results (Tables 8 and 9).

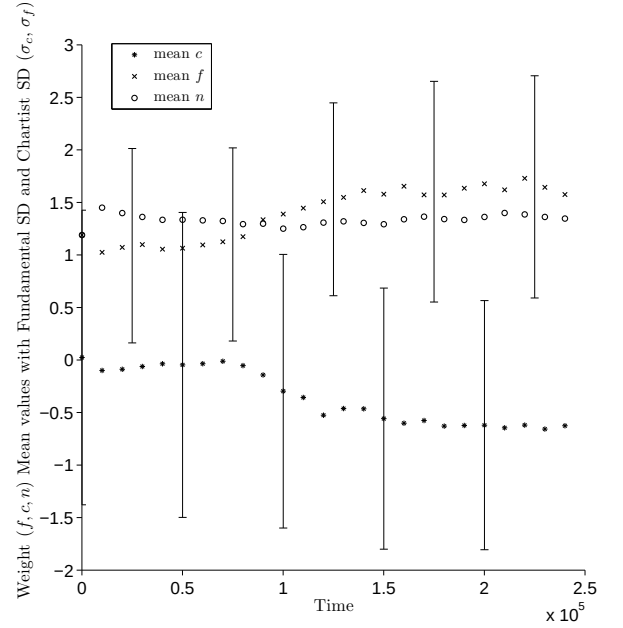


Figure 3: Contrarian Model: Mean Fundamentalist (f), Chartist (c) and Noise (n) weights with time and Fundamentalist and Chartist Standard Deviation (σ_f, σ_c)

6 Conclusion

While imitation may contribute to the generation of long memory phenomena in real financial markets other factors must play a role in producing stable long memory phenomena over time. Our model which incorporates contrarianism and strong disparity between strategies is able to generate a more dynamic behaviour in steady state, and is therefore able to produce more consistent long memory results across different free-parameter settings. The LY model [LeBaron and Yamamoto, 2007; 2008] is not stable with respect to variation in free-parameter settings and model execution time. This was caused by the convergence of the Genetic Algorithm to a smaller and smaller strategy space and a loss, therefore, of the dynamic that causes the long memory phenomena. We modify the LY model adding atomic agents and increased contrarianism (the Contrarian Model) which retains the dynamic necessary to generate stable long memory phenomena (Section 5).

We conjecture that any model that causes and maintains a dynamic switching behaviour will produce stable long memory and that a non-learning heuristic model would also produce positive long memory phenomena; this is the subject of future research.

References

- [Cont, 2001] R. Cont. Empirical Properties of Asset Returns: Stylised Facts and Statistical Issues. *Quantitative Finance*, 1:223–236, 2001.
- [Conrad, 1998] R. Cont. An Anatomy of Trading Strategies. *The Review of Financial Studies*, 11(3):489–519, 1998.

- [Ding *et al.*, 1993] Z. Ding, C. W. J. Granger, and R. F. Engle. A Long Memory Property of Stock Market Returns and a New Model. *Journal of Empirical Finance*, 1:83–106, 1993.
- [Engle, 1982] R. F. Engle. Autoregressive Conditional Heteroskedasticity with Estimates of the Variance of United Kingdom Inflation. *Econometrica*, 50:987–1008, 1982.
- [Evans and Honkapohja, 2001] G. W. Evans and S. Honkapohja. *Learning and Expectations in Macroeconomics*. Princeton University Press, 2001.
- [Heemeijer *et al.*, 2004] P. Heemeijer, C. H. Hommes, J. Sonnemans, and J. Tuinstra. Forming Price Expectations in Positive and Negative Feedback Systems. Technical Report 04-15, Universiteit van Amsterdam, 2004.
- [Lamer, 1990] C. G. Lamoureux and W. D. Lastrapes. Heteroskedasticity in Stock Return Data: Volume Versus GARCH Effects. *Journal of Finance*, 45(1):221–229, 1990.
- [LeBaron and Yamamoto, 2007] B. LeBaron and R. Yamamoto. Long-memory in an Order-Driven Market. *Physica A*, 383:85–89, 2007.
- [LeBaron and Yamamoto, 2008] B. LeBaron and R. Yamamoto. The Impact of Imitation on Long-Memory in an Order Driven Market. *Eastern Economic Journal*, 34:504–517, 2008.
- [Lillo *et al.*, 2005] F. Lillo, S. Mike, and J. D. Farmer. Theory for Long Memory in Supply and Demand. *Physical Review E*, 7106:287–297, 2005.
- [Lo, 1991] A. W. Lo. Long-Term Memory in Stock Market Prices. *Econometrica*, 59:1279–1314, 1991.
- [Lobato and Velasco, 2000] I. Lobato and C. Velasco. Long Memory in Stock-Market Trading Volume. *Business and Economic Statistics*, 18(4):410–426, 2000.
- [Mantegna and Stanley, 1997] R. N. Mantegna and E. H. Stanley. Stock Market Dynamics and Turbulence: Parallel Analysis of Fluctuation Phenomena. *Physica A*, 239(1-3):255–266, 1997.
- [Mantegna and Stanley, 2002] R. N. Mantegna and E. H. Stanley. Scaling Behavior in the Dynamics of an Economic Index. *Nature*, 376:46–49, 2002.
- [Pagan, 1996] A. Pagan. The Econometrics of Financial Markets. *Journal of Empirical Finance*, 3(1):15–102, 1996.

Self-Confirming Price Prediction for Bidding in Simultaneous Second-Price Sealed-Bid Auctions

Dong Young Yoon and Michael P. Wellman

University of Michigan
Ann Arbor, MI 48109-2121, USA
{dyoon,wellman}@umich.edu

Abstract

Bidding strategies for simultaneous interdependent markets typically rely on price predictions to manage uncertainty across markets. Previous work on simultaneous ascending auctions has found *self-confirming* price predictions particularly effective in dealing with the exposure problem. We adapt this approach to a setting with simultaneous one-shot (SPSB) auctions, producing self-confirming prediction versions of a range of heuristic strategies previously proposed in the literature. An in-depth simulation study in service of empirical game-theoretic analysis finds strong support for the BidEvaluator family of strategies in this setting.

1 Introduction

One of the most attractive features of automated trading is the ability to monitor and participate in many markets simultaneously. Compared to human traders, a software agent can take in data from multiple sources at high throughput, and in principle process a massive quantity of information relevant to trading decisions in short time spans. Dealing with multiple markets also poses one of the greatest strategic challenges for automated trading. When markets interact, a strategy for bidding in one market must consider the implications of and ramifications for outcomes in others.

Markets are *interdependent* when an agent's preference for the outcome of one market depends on the outcomes in other markets. For instance, when an agent's value for one good is increased by obtaining another, the goods are *complementary*. Dealing with multiple markets under complementary preferences presents an agent with the classic *exposure problem*: to obtain a valuable bundle, the agent must risk getting stuck with a strict subset of the goods, which it may not have wanted at the prevailing prices. For example, in the Trading Agent Competition (TAC) Travel game [Wellman et al., 2007], agents face an exposure problem for hotels—they must obtain a room each night for the client, otherwise the whole trip is infeasible.

Experience from TAC and many other domains has demonstrated the importance of *price prediction* for bidding in interdependent markets [Wellman et al., 2004]. Given probabilistic projections of prices across markets, the agents can man-

age exposure risk, choosing bids that optimally trade off the profits and losses of the possible bundles of goods obtained.

Greenwald and Boyan [2004] framed the problem of bidding across interdependent markets given probabilistic price predictions. Follow-on work [Greenwald et al., 2009, Wellman et al., 2007] formalized this bidding problem in decision-theoretic terms, and established properties of optimal bidding strategies given the assumption that bids do not affect other-agent behaviors. Further experimental comparison was performed by Greenwald et al. [2010]. These works introduced a taxonomy of heuristic bidding strategies, which we employ in the current study as described in §3.

The current study extends or departs from this previous line of investigation in three ways:

1. The market mechanism is simultaneous one-shot auctions, specifically second-price sealed bid auctions. This contrasts with the abstract “pseudo-auction” model employed in theoretical analysis, and forms of ascending auction employed in experimental studies.
2. Our analysis is based on an empirical game model derived through simulation of strategy profiles.
3. The bidding strategies employ market predictions based on derivation of *self-confirming prices* (defined in §4).

The idea of bidding based on self-confirming prices was first explored in the context of simultaneous ascending auctions (SAAs) [Cramton, 2005]. For the SAA environment, we found bidding using such predictions to be highly effective for dealing with the exposure problem [Wellman et al., 2008]. Given this success, the approach is a natural candidate for that role in simultaneous one-shot auctions. To our knowledge, no other general price-prediction methods have been proposed for simultaneous one-shot auctions, other than learning from historical observations.

2 Environment

Our analysis environment is defined by a market mechanism, and a distribution of valuations over goods ascribed to trading agents. The strategies defined here are applicable for any valuation distribution, but our experimental results are based on the particular distribution specified here.

2.1 Market Mechanism: Simultaneous SPSB

The *second-price sealed-bid* (SPSB) auction is one of the classic basic auction mechanisms analyzed in the seminal work of Vickrey [1961]. That the bids are sealed means that bidders have no information about the bids of other participants until the outcome is revealed. That it is second-price means that the winning bidder pays the second-highest bid rather than its own (highest) bid.

We consider a situation where multiple goods are auctioned through simultaneous SPSB auctions. The mechanism is *one-shot*, that is, the agents submit bids to each auction in parallel, upon which the auctions compute and report their results.

Our study employs the familiar *independent private values* model [Krishna, 2010], where each agent’s valuation for outcomes is drawn independently from a common-knowledge probability distribution. For a single SPSB auction with independent private values, it is a dominant strategy for an agent to bid its true valuation. This fails to hold for multiple simultaneous SPSB auctions, unless the agents’ valuations across goods are also independent. When agents’ valuations for different goods interdepend (e.g., through complementarity), bidding truthfully in simultaneous individual-good auctions is not even an option, as the value for an individual good is not well-defined.

2.2 Valuations: Market-based Scheduling

Let $\mathcal{X}$ be the set of goods available in all markets. The valuation function $v_i : 2^{\mathcal{X}} \rightarrow \mathbb{R}$ specifies agent i ’s value for the subsets of goods it may obtain in the markets.

The valuation distribution we employ in this study is based on a model of *market-based scheduling* [Reeves et al., 2005]. In this model, goods represent time slots of availability for some resource: for example, a machine, a meeting room, a vehicle, or a skilled laborer. Agents have tasks, which require this resource for some duration of time to complete.

Specifically, let there be n agents, and $\mathcal{X} = \{x_1, \dots, x_m\}$ the set of m time slots available to be scheduled. Agent i ’s task requires λ_i time slots to accomplish, and the agent values a set of time slots according to when they enable completion of the task. If an agent i acquires λ_i time slots by the deadline t , it obtains value v_i^t . The value with respect to the deadline is a nonincreasing function: for all i , if $t < t'$ then $v_i^t \geq v_i^{t'}$. If it fails to obtain a sufficient set of goods, the agent accrues value $v_i^\infty = 0$. Let $X \subseteq \mathcal{X}$ denote a set of slots. The expression $|\{x_j \in X \mid j \leq t\}|$ represents the number of these that are for time t or earlier. We can thus write

$$T(X, \lambda) = \min(\{t \text{ s.t. } |\{x_j \in X \mid j \leq t\}| \geq \lambda\} \cup \{\infty\})$$

to denote the earliest time by which X contains at least λ slots. The overall valuation function for agent i is then

$$v_i(X) = v_i^{T(X, \lambda_i)}.$$

In our experiments, we take $n = 8$ and $m = 5$. For each agent, a task length λ_i is drawn uniformly over the integers $\{1, \dots, 5\}$. The deadline values are drawn uniformly over $\{1, \dots, 50\}$, then pruned to impose monotonicity [Reeves et al., 2005]. The valuations induced by this scheduling scenario exhibit strong complementarity among goods. When

$\lambda > 1$, the agent gets no value at all for goods in a bundle of fewer than λ . On the other hand, there is some degree of substitutability across goods when there may be multiple ways of acquiring a bundle of the required size.

3 Bidding with Price Predictions

To deal effectively with interdependent markets, an agent’s bid in each auction must reflect its beliefs about the outcomes of others. Following prior work, we presume agents have some source of *predictions* about the prices at which it might obtain goods in the respective auctions. The bidding strategy is then a function of the available price prediction. In this section, we present the bidding strategies considered in this study, along with relevant concepts necessary to understand how the strategies operate. Our methods for *deriving* price predictions are presented in the following section.

3.1 Marginal Values and Optimal Bundles

Interdependence dictates that the value of individual goods be assessed relative to the bundle of goods that an agent obtains. This can be expressed by the notion of *marginal value*.

Definition 1 (Marginal Value) Agent i ’s *marginal value*, $\mu_i(x, X)$, for good x with respect to a fixed bundle of other goods X is given by:

$$\mu_i(x, X) = v_i(X \cup \{x\}) - v_i(X).$$

Given a fixed vector of prices, $\mathbf{p} = \langle p_1, \dots, p_m \rangle$, let $\sigma_i(X, \mathbf{p})$ denote agent i ’s surplus from obtaining the set of goods X at those prices:

$$\sigma_i(X, \mathbf{p}) \equiv v_i(X) - \sum_{j|x_j \in X} p_j. \quad (1)$$

Definition 2 (Acquisition [Boyan and Greenwald, 2001]) Given prices $\mathbf{p}$, the acquisition problem selects optimal goods to purchase:

$$X^* = \text{ACQ}_i(\mathbf{p}) \equiv \arg \max_{X \subseteq \mathcal{X}} \sigma_i(X, \mathbf{p}),$$

where σ is defined by (1).

Faced with known prices, an optimal strategy would be to compute a solution $X^* = \text{ACQ}(\mathbf{p})$ and buy the goods X^* . Its result is the optimal surplus at these prices:

$$\sigma_i^*(\mathbf{p}) \equiv \sigma_i(\text{ACQ}_i(\mathbf{p}), \mathbf{p}).$$

To assess goods with respect to given prices, we extend the concept of marginal value. Let $\mathbf{p}[p_j \leftarrow q]$ be a version of the price vector with the j th element revised as indicated: $\mathbf{p}[p_j \leftarrow q] = \langle p_1, \dots, p_{j-1}, q, p_{j+1}, \dots, p_m \rangle$.

Definition 3 (Marginal Value with Buying Opportunities) Agent i ’s marginal value $\mu_i(x_j, \mathbf{p})$ for good x_j with respect to prices $\mathbf{p}$ is given by:

$$\mu_i(x_j, \mathbf{p}) = \sigma_i^*(\mathbf{p}[p_j \leftarrow 0]) - \sigma_i^*(\mathbf{p}[p_j \leftarrow \infty]). \quad (2)$$

Here, $\sigma_i^*(\mathbf{p}[p_j \leftarrow 0])$ represents agent i ’s optimal surplus (solving its acquisition problem) at the given prices, assuming it receives good x_j for free. Similarly, $\sigma_i^*(\mathbf{p}[p_j \leftarrow \infty])$ represents the optimal surplus if x_j were unavailable. The

difference is precisely the marginal value of good x_j with respect to its buying opportunities for other goods. Note that Definition 3 can be viewed as a generalization of Definition 1, under the interpretation that goods in X have zero price, and all other goods have infinite price.

3.2 A Non-Predictive Baseline Strategy

To calibrate the performance of bidding strategies based on price prediction, we introduce a baseline strategy that employs no price information. Without price guidance, there is little basis to choose among alternative bundles in a one-shot auction. We therefore simply assume the agent selects its most valued set of goods, which is the solution to its acquisition problem at zero prices. Let $X^* = \text{ACQ}_i(\mathbf{0})$.

The strategy **BaselineBidding** bids for goods in X^* , with a total bid budget of $v(X^*)$. To determine the bid price for each individual good, it starts with the value for each good taken individually, and evenly divides the remaining budget among the goods. More precisely, let the excess value be given by

$$e = v(X^*) - \sum_{x \in X^*} v(\{x\}).$$

For complementary preferences, e is guaranteed to be non-negative. **BaselineBidding** then bids for good j as follows:

$$\text{BaselineBidding}_j = \begin{cases} v(\{x_j\}) + \frac{e}{|X^*|} & \text{if } x_j \in X^*, \\ 0 & \text{otherwise.} \end{cases}$$

For the market-based scheduling valuation, X^* comprises the first λ_i goods. Bundles with additional goods also maximize value, but we take the minimal maximizing set. Individual goods are valued at zero, unless $\lambda_i = 1$, in which case $v_i(x_j) = v_i^j$. **BaselineBidding** agents with a single-unit task duration thus bid true value for the first slot, and otherwise they bid for the first λ_i goods at the price $v_i^{\lambda_i}/\lambda_i$.

3.3 Bidding with Point Price Predictions

We consider first a set of strategies that employ predictions in the form of a vector of point prices, $\mathbf{p} = \langle p_1, \dots, p_m \rangle$. All strategies considered in this section and the next are described with further motivation and detail by Wellman et al. [2007, Chapter 5].

Like **BaselineBidding**, strategies in the **TargetBidder** family restrict their bidding to goods in $X^* = \text{ACQ}_i(\mathbf{p})$. They differ in the amounts they bid. The **TargetPrice** strategy bids the predicted price for each good in its optimal bundle.

$$\text{TargetPrice}_j = \begin{cases} p_j & \text{if } x_j \in X^*, \\ 0 & \text{otherwise.} \end{cases}$$

TargetMV bids at marginal value rather than predicted price.

$$\text{TargetMV}_j = \begin{cases} \mu(x_j, \mathbf{p}) & \text{if } x_j \in X^*, \\ 0 & \text{otherwise.} \end{cases}$$

TargetMV* also bids at marginal value, except that it calculates μ under the assumption that goods outside the target bundle are unavailable.

$$\text{TargetMV}^*_j = \begin{cases} \mu(x_j, \mathbf{p}[p_\ell \leftarrow \infty \mid x_\ell \notin X^*]) & \text{if } x_j \in X^*, \\ 0 & \text{otherwise.} \end{cases}$$

The virtue of **TargetBidder** strategies is that they will not obtain goods outside the desired bundle. A weakness is their fragility—if the agent fails to get the full bundle, there is no recourse to others outside the set. The **StraightMV** strategy hedges against incorrect predictions by bidding marginal value for all goods.

$$\text{StraightMV}_j = \mu(x_j, \mathbf{p}).$$

3.4 Bidding with Price Distributions

A point price estimate fails to convey the uncertainty inherent in predicting the prices at which goods will be available. Probability distributions over prices provide a more general representation, expressing degrees of belief over the possible prices that might obtain. We therefore expand our strategy set to incorporate bidding methods that take as input F , a probability distribution over prices.

Expected Value Methods

The *expected value method* [Birge and Louveaux, 1997] approximates a stochastic optimization method by collapsing probability distributions into point estimates through expectation. Let $\hat{\mathbf{p}}_F = \langle \hat{p}_1, \dots, \hat{p}_m \rangle$, where $\hat{p}_j = \mathbb{E}_F[p_j]$ is the expectation of p_j under distribution prediction F .

Any bidding strategy defined for point price predictions can be adapted to take price distributions through the expected value method, simply by using $\hat{\mathbf{p}}_F$ for the point price prediction. We thus define **StraightMU**, **TargetMU**, and **TargetMU*** as distribution-based methods that employ corresponding point prediction strategies in this way. “MU” stands for “marginal utility”, but its usage here is simply to distinguish the name from its corresponding point strategy ending with “MV”. Our implementation calculates $\hat{\mathbf{p}}_F$ by sampling from F . Since accuracy depends on the number k of samples taken, we indicate a version of the strategy by appending “ k ” to the name. For example, **StraightMU8** calculates $\hat{\mathbf{p}}_F$ as the mean of eight samples from F , and employs **StraightMV** with that average price vector as input.

AverageMU

Whereas **StraightMU** bids the marginal value of the expected price, the strategy **AverageMU** bids the expected marginal value:

$$\text{AverageMU}_j = \mathbb{E}_F[\mu(x_j, \mathbf{p})].$$

Our implementation samples from the price distribution, calculating marginal values from each sample, and averaging the results.

BidEvaluator

The final approach we consider is to generate candidate bid vectors, and evaluate them according to the given price distribution. The **BidEvaluator** strategy uses other bidding strategies to propose candidates, and estimates their performance by sampling from F and averaging over the resulting surpluses. For instance, **BidEvaluator(SMU8)** generates candidates using **StraightMU8**. Since each invocation of **StraightMU8** employs a new draw of eight samples from F to estimate $\hat{\mathbf{p}}_F$, we generally obtain different bids. In our study, we define **BidEvaluator(SMU8)** to generate four candidate bids using **StraightMU8**, which takes 32 samples from

F . We employ another 32 samples to evaluate these candidates, for a total of 64 required by this strategy. The result of BidEvaluator(SMU8) is the candidate bid vector which performed best on average according to the 32 test samples.

4 Self-Confirming Price Predictions

Now that we have a suite of strategies that employ price predictions, we turn next to the question of how to generate such predictions. We propose here to employ *self-confirming price predictions*, originally introduced and evaluated in the context of simultaneous ascending auctions [Wellman et al., 2008]. Our definitions and derivation methods follow this prior work.

In the following definitions, let Γ be an instance of a simultaneous auction game, and PP denote a bidding strategy that employs price predictions: either points or distributions.

Definition 4 (Self-Confirming Price Prediction) *The prediction π is a self-confirming prediction for strategy PP in Γ iff π is equal to the expectation (over the type distribution) of the final prices when all agents play $PP(\pi)$. Similarly, F is a self-confirming price distribution for PP in Γ iff F is equal to the expectation (over the type distribution) of the final prices when all agents play $PP(F)$.*

We also consider a relaxed version of self-confirming distributions, where the final prices and predictions need coincide only on marginal—rather than joint—probabilities.

4.1 Procedure for Self-Confirming Point Price Prediction Approximation

We employ a simple iterative procedure to derive an approximate self-confirming point prediction for strategy PP in Γ .

Our procedure is defined by the following set of parameters:

1. An initial price prediction π^0 . We take $\pi^0 = \langle 1, \dots, 1 \rangle$.
2. A number G of game instances per iteration. We set G to one million in our study.
3. A threshold δ . When the distance between results of successive iterations is less than δ , we halt the procedure and return the latest result. We set this at 0.001 times the maximum value, hence $\delta = 0.05$.
4. A maximum number L of iterations. We take $L = 100$ in our experiments.
5. A decaying sequence, κ_t , which controls the updating of predictions in each iteration. We employ $\kappa_t = \frac{L-t+1}{L}$.

At iteration t , we run G instances of game Γ , with all agents playing strategy $PP(\pi^{t-1})$. We tally the resulting prices resulting from each instance, and denote the average price vector by $\bar{p}^t$. We then update the price prediction by

$$\pi^t \leftarrow \pi^{t-1} + \kappa_t (\bar{p}^t - \pi^{t-1}).$$

When the maximum price difference between two successive iterations is below threshold,

$$\max_j |\pi_j^t - \pi_j^{t-1}| \leq \delta,$$

the procedure halts and returns π^t .

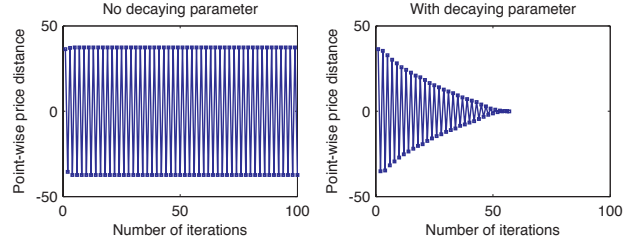


Figure 1: Effect of the decay parameter for iterative update of TargetMV* predictions.

We evaluated the convergence of this procedure with respect to three different PP strategies: StraightMV, TargetMV, and TargetMV*. For StraightMV, we reach a fixed point within 20 iterations, even with the setting $\kappa_t = 1$. For TargetMV and TargetMV*, the price predictions tend to oscillate, hence we need a decaying sequence to settle the process. For these cases, we do not expect the results to approximate self-confirmation as well. Nevertheless, the iterative procedure produces a well-defined prediction, which in many cases is a reasonable basis for bidding.

Figure 1 shows the series of pointwise price distances between consecutive iterations in deriving point price predictions for TargetMV* with and without a decay parameter. The absence of a decay parameter is expressed simply by setting $\kappa_t = 1$. At this setting, the iterative update of price predictions for TargetMV* fails to converge. Since TargetMV* bids for goods in its target bundle under the assumption that no others are available, these bids tend to be at the extremes of its value range. With all agents playing this strategy, this leads to high prices, which causes changes in targets in the next iteration (by all agents), causing these same goods to have very low prices then. TargetMV produces a similar pattern, though attenuated by its consideration of alternative buying opportunities. With regular oscillation, the effect of the decay sequence is to return the average of the swings that would otherwise be observed.

4.2 Procedure for Self-Confirming Price Distribution Approximation

A similar iterative procedure is used to derive self-confirming price distributions. The cumulative price distribution at iteration t is denoted F^t . We employ a discrete approximation, rounding all observed prices to the closest integer. We can thus represent the distribution using a probability mass function, $f(q)$, corresponding to the probability $\Pr(p = q)$. We adopt the simplifying (and incorrect) assumption that prices are probabilistically independent across goods, which allows us to maintain this as an array of marginal distributions, $\langle f_1(q_1), \dots, f_m(q_m) \rangle$.

To measure the difference between distributions at successive iterations, we adopt the Kolmogorov-Smirnov (KS) statistic,

$$KS(F, F') = \max_x |F(x) - F'(x)|.$$

Since we maintain our prediction in terms of marginal distributions, our comparison takes the maximum of the KS statis-

tic separately for each good: $KS_{\text{marg}} = \max_j KS(F_j, F'_j)$.

Our initial prediction F^0 considers all integer prices in the feasible range equally likely. As for the point-prediction case, at iteration t , we run G instances of game Γ . Here all agents play the distribution-prediction strategy $PP(F^{t-1})$. We tally the resulting prices resulting from each instance, and denote the sample distribution for good j by $\bar{f}_j^t$. We then update the distribution prediction, for all j and price possibilities q , by

$$f_j^t(q) \leftarrow f_j^{t-1}(q) + \kappa_t (\bar{f}_j^t(q) - f_j^{t-1}(q)).$$

When $KS_{\text{marg}}(F^{t-1}, F^t)$ is below threshold (here $\delta = 0.01$), we halt and return F^t .

A comparison of the self-confirming price predictions derived for StraightMU (distributions) and StraightMV (points) is depicted in Figure 2.

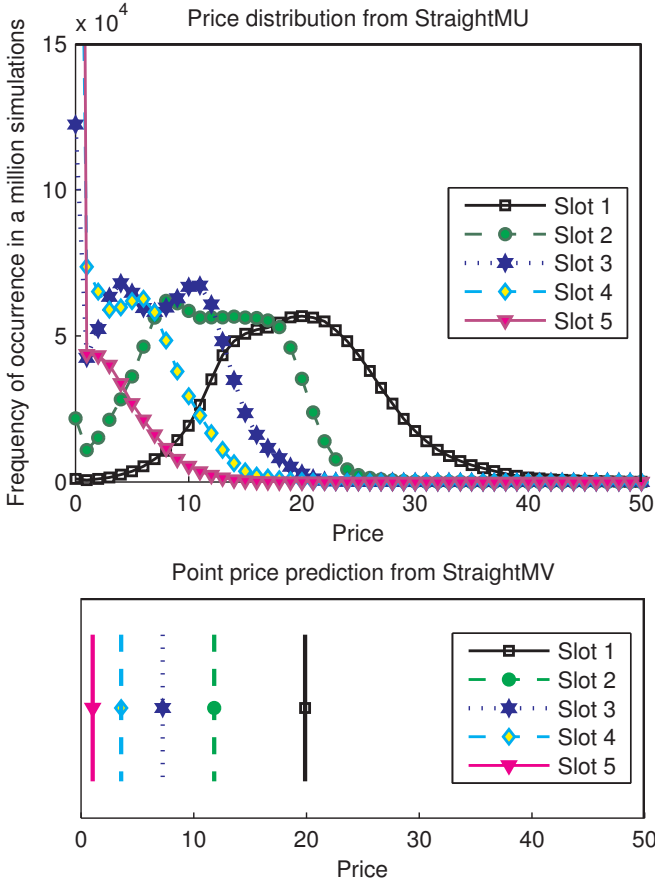


Figure 2: Self-confirming price predictions derived for StraightMU and StraightMV for a particular game.

4.3 Accuracy of Self-Confirming Price Predictions

The decay parameter ensures the iterative procedure for deriving price predictions will terminate, but potentially masks violations of self-confirmation in the result. To test the accuracy of self-confirming price predictions for this study, we ran an extra iteration of one million game instances after a supposed self-confirming price prediction is derived by the

procedure. The distance of derived price predictions from the resulting prices (without any attenuation) are reported in Table 1. As shown in the table, all are self-confirming to a fine level of approximation.

Point Strategy	Max Distance
StraightMV	0.0116
TargetMV	0.0660
TargetMV*	0.0553
Distribution Strategy	KS Statistic
StraightMU64	0.00450
TargetMU64	0.00879
TargetMU*64	0.00788
AverageMU64	0.00516

Table 1: Accuracy of self-confirming point and distribution price predictions.

5 Empirical Game-Theoretic Analysis of Bidding Strategies

We evaluate the performance of the bidding strategies with self-confirming price predictions through a process of *empirical game-theoretic analysis* (EGTA). In this EGTA approach, we simulate profiles of an enumerated strategy set, and use the observed payoffs to estimate a normal-form game. The induced game model supports conclusions about the strategic properties of subject strategies and profiles, through application of standard solution concepts.

5.1 Hierarchical Game Reduction

The space of strategy profiles grows exponentially in the number of players and strategies considered. Specifically, for a symmetric normal-form game with n players and S strategies, there are $\binom{n+S-1}{n}$ distinct strategy profiles. The primary scenario we analyze includes eight bidding agents. To enable us to scale up the strategy space, we employ *hierarchical game reduction* [Wellman et al., 2005] to recast the scenario as a four-player game. In this framing, each player controls two bidding agents, imposing its selected strategy on each. The reduction entails some loss in fidelity, but we have found in many EGTA studies that it provides a good approximation at exponentially reduced computational expense.

5.2 Regret

We evaluate the stability of strategy profiles by measuring *regret*, the maximal gain a player could achieve by deviating from the profile. Formally, let $\Gamma = \{n, S, u(\cdot)\}$ be a symmetric normal-form game with n players, strategy space S for each player (since the game is symmetric), and payoff function $u : S \times S^{n-1} \rightarrow \mathbb{R}$. The expression $u(s_i, s_{-i})$ represents the payoff to playing strategy s_i in a profile where the other players play strategies $s_{-i} \in S^{n-1}$.

Definition 5 (regret) The regret $\epsilon(s)$ of a strategy profile $s = (s_1, \dots, s_n)$ is given by

$$\epsilon(s) = \max_i \max_{s'_i \in S} (u(s'_i, s_{-i}) - u(s_i, s_{-i})).$$

A Nash equilibrium profile has zero regret, and more generally regret provides a measure of approximation to equilibrium. Using this regret definition, profile s is an $\epsilon(s)$ -Nash equilibrium.

Regret is a property of profiles. Evaluation of a particular strategy is inherently relative to a context of strategies played by other agents. Jordan et al. [2007] proposed ranking strategies according to their performance when other agents are playing an equilibrium.

Definition 6 (NE regret [Jordan, 2010]) Let s^{NE} be a symmetric (possibly mixed) Nash equilibrium of game Γ . The NE regret of a strategy $s_i \in S$ is $u(s_i^{NE}, s_{-i}^{NE}) - u(s_i, s_{-i}^{NE})$.

NE regret represents the loss of an agent deviating to a specified strategy from a Nash equilibrium of a game. The rationale for this measure comes from the judgment that all else equal, Nash equilibria provide a most compelling strategic context for evaluating a given strategy.

5.3 Simulations

Our game is defined by the scheduling valuation distribution described in §2.2, with $n = 8$ agents and $m = 5$ time slots. We simulate a game instance for a specified strategy profile by drawing from the valuation distribution, applying the strategies to generate bids, and calculating the outcome with simultaneous one-shot SPSB auctions. Since each instance is so computationally simple, we sample one million instances per profile for estimating expected payoffs in all experiments reported here.

5.4 Effect of Price Samples

As a preliminary study, we investigated the effect of the number of price samples on the performance of strategies based on distribution predictions. Payoffs for an 8-player empirical game over two strategies, StraightMU8 or StraightMU64, are displayed in Figure 3. Recall that these strategies differ only in the number of price samples that are averaged to produce the price vector employed in marginal value calculations. It is apparent from the graph that StraightMU64 dominates StraightMU8, as we might expect.

Additional price sampling, however, produces diminishing returns. We estimated the 4-player empirical reduced game for five versions of StraightMU, differing on how much they sample prices. Table 2 presents some relevant statistics. Although StraightMU64 and StraightMU256 are dominated, the other strategies are all present in some equilibrium. The performance gain after around 1000 samples becomes difficult to measure.

Number of samples (StraightMU)	Avg. payoff of symm. profile	Regret of symm. profile
64	0.434	0.048
256	0.483	0.024
1024	0.506	0.015
4096	0.519	0
8192	0.526	0

Table 2: StraightMU with varying number of price samples.

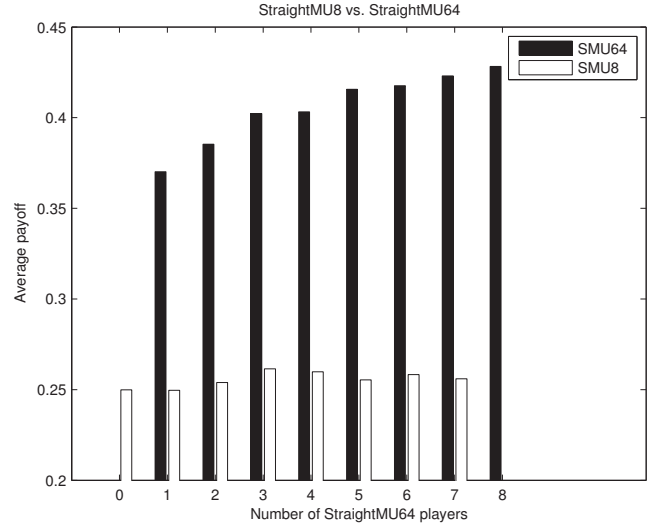


Figure 3: StraightMU8 vs. StraightMU64

5.5 Strategy Space

To balance performance and computational cost, we choose a sample size of 64 for the price distribution strategies in our primary empirical game analysis. To ensure fairness, our BidEvaluator strategy also uses 64 samples in total, as explained in §3.4.

We analyze an empirical game that consists of eleven strategies: one BaselineBidding, three point price predictors—StraightMV, TargetMV, and TargetMV*, and seven distribution predictors—StraightMU64, TargetMU64, TargetMU*64, AverageMU64, BidEvaluator(TMU8), BidEvaluator(TMU*8), and BidEvaluator(SMU8). Seven of these employ self-confirming price predictions derived by the iterative process of §4 with the strategy itself playing the role of PP. The exceptions are BaselineBidding, which employs no price prediction, and the three BidEvaluator strategies, which use price distributions that are self-confirming with respect to other specified bidding strategies. We follow the main analysis by investigating a few smaller empirical games with additional strategies suggested by our initial findings.

5.6 Primary Game Results

We evaluated the payoffs for each of the 1001 distinct strategy profiles of our symmetric 4-player game with eleven strategy choices. The strategy BidEvaluator(SMU8) emerged as the clear choice—in fact it is a dominant strategy in this game. Table 3 lists all of the strategies, and for each presents the regret of the profile where all agents follow that strategy. Since BidEvaluator(SMU8) is dominant, the regret in all cases represents the benefit of deviating to that strategy. The strategies are ranked by NE regret (Definition 6), which in this case is the regret of the profile where one player employs the corresponding strategy and the rest play BidEvaluator(SMU8).

One perhaps surprising observation is that a point prediction strategy, TargetMV*, is runner-up with respect to the NE-regret measure. Prior literature had found it effective

Strategy	Regret of symmetric profile	NE Regret	Eff. (%)
BidEvaluator(SMU8)	0	0	77.5
TargetMV*	0.393	0.351	78.2
TargetMU*64	0.590	0.438	76.0
AverageMU64	0.783	0.450	72.2
BidEvaluator(TMU*8)	1.099	0.665	74.0
StraightMV	0.648	0.684	73.8
StraightMU64	0.720	0.726	72.9
TargetMV	1.023	0.990	67.1
BidEvaluator(TMU8)	1.179	1.055	77.5
TargetMU64	1.161	1.059	65.5
BaselineBidding	2.164	2.890	60.5
TargetMU*256	0.452	0.350	77.2
AverageMU256	0.751	0.431	72.9
StraightMU256	0.653	0.677	73.6

Table 3: Regret and efficiency of bidding strategies in the original game, and the follow-up game with 256-sample strategies.

among point predictors, but the fact that it outperforms many distribution predictors is worth some attention in future work to explain.

We also report the *allocative efficiency* achieved by the market with these bidding strategies. For each game instance, we calculate the allocation of time slots to agents that maximizes the total value obtained by all agents. The outcomes of symmetric strategy profiles are then compared with the optimal distribution to yield a measure of efficiency. As illustrated in Table 3, the equilibrium strategy achieves 77.5% of the social optimum. Several others are in that ballpark, but BaselineBidding and the unstarred TargetMV/TargetMU strategies perform substantially worse.

To test the robustness of BidEvaluator(SMU8)’s dominance, we extended our analysis to include instances of the most successful distribution predictors, but with a four-fold increase in their allotted price samples. We constructed a follow-up empirical game consisting of BidEvaluator(SMU8) and three distribution-based strategies: StraightMU256, TargetMU*256, and AverageMU256. We found that BidEvaluator(SMU8) is also a dominant strategy in this game. The regret and efficiency values for these strategies (presented at the bottom of Table 3) are incrementally better than their 64-sample counterparts, but the ordering among these is unchanged. Note that with the additional samples, TargetMU*256 achieves par with its point-predicting counterpart, TargetMV*.

5.7 Hybrid Strategies

From the analysis thus far we cannot separate the effects on performance due to accurate price prediction from that of exploiting a given prediction well in bidding. With the exception of BidEvaluators, all strategies employ predictions that are self-confirming with respect to themselves. It is also possible to construct hybrid bidders, which employ different strategies for purposes of deriving predictions and for pur-

poses of bidding given predictions.

To explore this idea, we considered a representative selection of strategies that mix-and-match price prediction and bidding strategy flavors. Our next empirical game starts with BidEvaluator(SMU8), and adds four strategies that mix across StraightMV/StraightMU and TargetMV*/TargetMU*, and two BidEvaluators that employ different methods for generating predictions and candidate bids. To identify a hybrid strategy we adopt the naming convention *BiddingStrategy.PredictionStrategy*. For instance, StraightMU64.TMU*64 bids with StraightMU64 using the price prediction derived from TargetMU*64. The strategy BidEvaluator(TMU*8).SMU8 generates bid candidates using TargetMU*8, but evaluates them using the self-confirming prediction of StraightMU8. We also introduced BidEvaluatorMix, which is identical to BidEvaluator(SMU8) except that two of its four candidate bids are generated using TargetMU*8.

Table 4 ranks the eight strategies of this empirical game by NE regret. The NE employed for this ranking is a symmetric mixture of BidEvaluator(SMU8) at probability 0.507, and BidEvaluatorMix8 at 0.493. There is also a pure strategy Nash equilibrium (PSNE) with two players choosing each of these strategies (efficiency: 78.4%). This suggests some benefit to diversifying the source of bid candidates, particularly when other agents use a single source.

Strategy	Regret of symm. profile	NE Regret
BidEvaluator(SMU8)	0.143	0
BidEvaluatorMix8	0.167	0
BidEvaluator(TMU*8).SMU8	0.757	0.153
TargetMV*.SMV	0.321	0.219
TargetMU*64.SMU64	0.425	0.267
BidEvaluator(SMU8).TMU*8	0.998	0.669
StraightMV.TMV*	1.214	0.825
StraightMU64.TMU*64	1.342	0.891

Table 4: Regret of hybrid bidding strategies.

According to the NE-regret measure, the TargetMV* and TargetMU* strategies benefit somewhat by hybridization. This is not very surprising, since the equilibrium comprises strategies that predict and predominantly bid based on StraightMU. For the same reason, it is logical that the StraightMV and StraightMU bidding strategies suffer by using predictions based on TargetMV or TargetMU strategies.

Most importantly, the result provides further supporting evidence for BidEvaluator strategies. The two strategies supported in equilibrium, as well as the next runner-up, are versions of BidEvaluator using distribution price predictions that are self-confirming for StraightMU.

5.8 Effect of Extending Available Samples

We investigated the potential advantage of allowing more computation (sampling) in the BidEvaluator(SMU) and BidEvaluatorMix strategies. There are three ways to use additional samples: marginal utility calculation, generating

additional bid candidates, or testing the candidates. Table 5 ranks by NE regret the combinations we evaluated in an empirical game. BidEvaluator(SMU64) with four candidate bids and 256 samples for testing and BidEvaluatorMix64 with the equivalent number of samples, where each strategy is chosen by two players, turn out to be the unique PSNE in this game. We observed a positive effect on performance with increased samples for marginal utility and testing, though the actual payoff benefit is relatively small. In fact, the performance of a strategy degrades with 1024 samples, demonstrating the limitation of extending available samples.

Strategy Base	C	T	Total samples	Regret of symm. profile	NE Regret
BEMix64	4	256	512	0.126	0
BE(SMU64)	4	256	512	0.246	0
BE(SMU64)	8	512	1024	0.254	0.017
BEMix64	8	512	1024	0.146	0.025
BE(SMU8)	4	32	64	0.323	0.112
BEMix8	4	32	64	0.291	0.157

Table 5: BidEvaluator(SMU) and BidEvaluatorMix variants with differing allotments of price samples (C = Candidate bids, T = Test samples).

6 Conclusion

We proposed the use of self-confirming price predictions for bidding in simultaneous SPSB auctions with complementary preferences. Simulations conducted for an 8-agent environment with scheduling valuations demonstrated the viability of this approach, and its applicability across a range of heuristic bidding strategies. An empirical game-theoretic analysis finds strong support for the BidEvaluator family of strategies, particularly in conjunction with price predictions that are self-confirming with respect to StraightMU bidders.

Further work will attempt to establish the generality of these conclusions, by exploring a wider range of market configurations: numbers of agents and goods, and valuation distributions. We also intend to explore the effects on BidEvaluator performance of extending computation to allow more price samples or bid candidates, and the possibility of employing price predictions that are self-confirming for BidEvaluator itself. Finally, we are quite interested in exploiting our ability to derive equilibria for these markets to assess the performance of alternative mechanisms. In particular, we should be able to compare the allocative efficiency of simultaneous one-shot versus ascending auctions, as well as simultaneous versus sequential market designs.

Acknowledgments

This research was supported in part by Grant CCF-0905139 from the U.S. National Science Foundation. We are grateful for helpful comments from Amy Greenwald, Eric Sodomka, and members of the Strategic Reasoning Group at the University of Michigan AI Laboratory.

References

- John Birge and Francois Louveaux. *Introduction to Stochastic Programming*. Springer, 1997.
- Justin Boyan and Amy Greenwald. Bid determination in simultaneous auctions: An agent architecture. In *Third ACM Conference on Electronic Commerce*, pages 210–212, Tampa, 2001.
- Peter Cramton. Simultaneous ascending auctions. In Peter Cramton, Yoav Shoham, and Richard Steinberg, editors, *Combinatorial Auctions*. MIT Press, 2005.
- Amy Greenwald and Justin Boyan. Bidding under uncertainty: Theory and experiments. In *Twentieth Conference on Uncertainty in Artificial Intelligence*, pages 209–216, Banff, 2004.
- Amy Greenwald, Seong Jae Lee, and Victor Naroditskiy. RoxyBot-06: Stochastic prediction and optimization in TAC travel. *Journal of Artificial Intelligence Research*, 36:513–546, 2009.
- Amy Greenwald, Victor Naroditskiy, and Seong Jae Lee. Bidding heuristics for simultaneous auctions: Lessons from TAC travel. In Wolfgang Ketter, Han La Poutré, Norman Sadeh, Onn Shehory, and William Walsh, editors, *Agent-Mediated Electronic Commerce and Trading Agent Design and Analysis*, volume 44 of *Lecture Notes in Business Information Processing*, pages 131–146. Springer-Verlag, 2010.
- Patrick R. Jordan. *Practical Strategic Reasoning with Applications in Market Games*. PhD thesis, University of Michigan, 2010.
- Patrick R. Jordan, Christopher Kiekintveld, and Michael P. Wellman. Empirical game-theoretic analysis of the TAC supply chain game. In *Sixth International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pages 1188–1195, Honolulu, 2007.
- Vijay Krishna. *Auction Theory*. Academic Press, second edition, 2010.
- Daniel M. Reeves, Michael P. Wellman, Jeffrey K. MacKie-Mason, and Anna Osepayshvili. Exploring bidding strategies for market-based scheduling. *Decision Support Systems*, 39:67–85, 2005.
- William Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *Journal of Finance*, 16:8–37, 1961.
- Michael P. Wellman, Daniel M. Reeves, Kevin M. Lochner, and Yevgeniy Vorobeychik. Price prediction in a trading agent competition. *Journal of Artificial Intelligence Research*, 21:19–36, 2004.
- Michael P. Wellman, Daniel M. Reeves, Kevin M. Lochner, Shih-Fen Cheng, and Rahul Suri. Approximate strategic reasoning through hierarchical reduction of large symmetric games. In *Twentieth National Conference on Artificial Intelligence*, pages 502–508, Pittsburgh, 2005.
- Michael P. Wellman, Amy Greenwald, and Peter Stone. *Autonomous Bidding Agents: Strategies and Lessons from the Trading Agent Competition*. MIT Press, 2007.
- Michael P. Wellman, Anna Osepayshvili, Jeffrey K. MacKie-Mason, and Daniel M. Reeves. Bidding strategies for simultaneous ascending auctions. *B. E. Journal of Theoretical Economics (Topics)*, 8(1), 2008.

Rank and Impression Estimation in a Stylized Model of Ad Auctions

Jordan Berg, Carleton Coffrin, Amy Greenwald, and Eric Sodomka

Department of Computer Science, Brown University

{jberg, cjc, amy, sodomka}@cs.brown.edu

Abstract

Empirical research on ad auctions often aims to estimate model parameters (e.g., click probabilities) from aggregate data (e.g., average positions, total impressions, etc.). This is not surprising, because aggregate data are what the search engines reveal. But it has recently been pointed out that aggregate data can produce biased estimates [1]. In this work, we construct a disaggregated model of positions and impressions from aggregate data. Previous work has demonstrated the feasibility and benefits of disaggregation [13]. We extend the previous approach by formulating the disaggregation process as two intertwined problems, *rank* and *impression estimation*. We solve these problems (and a natural simplification) using dynamic, mathematical, and constraint programming techniques. We then evaluate the merits of our solution techniques in a simulated ad auction environment, the Ad Auctions division of the annual Trading Agent Competition.

1 Introduction

Most Internet advertising space is sold via ad auctions, which operate as follows: Sometime in advance, a publisher (e.g., a search engine) solicits bids from advertisers on potential queries. Then, when a user submits one of those queries to a publisher, the user is shown a set of sponsored search results (i.e., ads) alongside so-called *organic* results based purely on relevance and importance, not on sponsorship. The placement (also called slot or position) of each ad is determined on-the-fly by the publisher, who runs an auction based on the previously submitted bids. Then, if/when the user clicks on an ad, the corresponding advertiser pays the publisher an amount determined by the auction—a so-called *cost-per-click* (CPC). Higher slots tend to cost more because they are more likely to generate clicks.

In this environment, both publishers and advertisers face difficult decision problems. The publishers must determine CPCs, and in what order to display the winning

ads, depending on bids. The advertisers in turn must decide what queries to bid on and the maximum CPC they are willing to pay. Correspondingly, work on sponsored search can be divided into two categories: ad auction design, which addresses the publishers’ problem (e.g., [2; 3; 5; 7; 12; 15]), and bidding strategy design, which addresses the advertisers’ problem (e.g., [4; 10; 13]). This paper falls into the latter category.

Adopting a common assumption in past research (e.g., [8; 13]),¹ we view the decision problem the advertisers face as decision-theoretic, not game-theoretic. As such, we assume the advertiser builds a model of the world around him, which predicts such things as the future bids and budgets of other advertisers and anticipated user behavior.² Then, taking this model as input, the advertiser makes its decisions.

Most existing empirical research on ad auctions aims to estimate models from aggregate data (e.g., average position, total number of impressions, etc.). This is not surprising, because aggregate data are what the search engines reveal. But it has recently been pointed out that aggregate data can produce biased estimates; for example, predicted click-through-rates can exceed actual [1]. Consequently, one must either figure out a way to adjust for potential bias in their estimates, or try to infer a disaggregated model of the intra-day variation in average positions, total number of impressions, etc. before estimating model parameters.

In this paper, we take the latter route. That is, we attempt to construct a disaggregated picture of slot dynamics over the course of a day—meaning which advertiser was in which slot when and for how long—from aggregate statistics. Previous work [13] has demonstrated the feasibility and benefits of disaggregation. We extend the previous approach by formulating the disaggregation process as two intertwined problems, *rank* and *impression estimation*. The impression estimation problem (IE) is to estimate the number of impressions that

¹See Wellman *et al.* [16] for a discussion of the validity of this assumption.

²In most auctions, the rules are known. In real-world ad auctions, however, the rules are often not known. Hence, in general, an advertiser should incorporate predictions about the auction rules themselves into its model of the world.

each bidder sees in each position. The rank and impression estimation problem (RIE) is to estimate each bidder’s rank as well, when sorted in (squashed) bid order. This decomposition is advantageous for several reasons, including that it allows us to clearly state and prove mathematical properties about *cascades*, the data structure we use to represent our solutions to these problems.

Using our theoretical insights, we develop a mathematical programming formulation of the rank and impression estimation problem, which can be solved with commercial integer programming solvers. We compare and contrast our mathematical programming formulation with a problem-specific tree search, similar to that which was developed previously [13]. We evaluate the merits of these two approaches using a simulated, experimental ad auction platform, the Ad Auctions division of the annual Trading Agent Competition (TAC AA) [9], for which both aggregate statistics and disaggregated data are readily available. In ongoing work, we are further evaluating the impact of these two competing approaches on our techniques for predicting things like opponents’ (squashed) bids and budgets.

This paper is organized as follows. In Sec. 2, we describe the stylized ad auctions model from which this work will be based. In Sec. 3, we define a data structure we call a *cascade*, which stores the number of impressions each advertiser sees in each slot. Further, we present a dynamic program (DP) that populates a cascade, given the total number of impressions seen by all advertisers and their ranks. We then state several key properties that characterize cascades. In Sec. 4, we define the rank and impression estimation problems, and we describe our integer and constraint programming solutions to IE, as well as a least discrepancy search which we wrap around these solutions to solve RIE. In Sec. 5, we report on the performance of our solutions in TAC AA relative to ground truth, which we calculate using our DP.

2 A Stylized Model

Imagine a set of N advertisers, each of which is interested in displaying its ad in one of M slots alongside the organic search results for a given query. At the start of each time period, each advertiser submits a bid and spending limit to the publisher. The submitted bids are then ranked by either bid or revenue, or something in between, as determined by a squashing parameter [11]. When a user searches for the given query, the ads that are displayed are those of the advertisers with the top M (squashed) bids, in order, from highest to lowest, of (squashed) bid. Advertisers pay the publisher as users click on their ads according to standard [7] generalized second price (GSP) auction rules (i.e., they pay the minimum amount they would have had to bid to maintain their assigned slot). Once an advertiser has paid the publisher so much that another click would put it over its spending limit, it drops out of all subsequent auctions, and the remaining advertisers are re-ranked. In particular, the advertisers previously ranked below that

advertiser move up one position in the ranking. This process continues until either all agents exhaust their budgets, or no further users place search queries.

At the end of the time period, each advertiser receives the following aggregate data from the publisher: its total number of impressions (i.e. the number of times its ad was displayed) and the average position of all advertisers that participated in the day’s auctions.

In the remainder of the paper, we use the term *agent* in place of advertiser or bidder.

3 Cascade Calculations

The first problem we present is a simplification of rank and impression estimation in which we know not only the rank of all agents, but the total number of impressions seen by each agent as well. We originally studied this problem because its solution is necessary to compute ground truth and is hence essential for experimentation with our constraint programming solvers for the IE and RIE problems. But in doing so, we identified mathematical properties of cascades which gave rise to integer programming solutions to IE and RIE.

3.1 The Cascade Data Structure

We represent the slot dynamics that arise in our stylized model using a data structure we call a *cascade*. In nature, a cascade is a structure in which water flows down a stair case shaped rock formation. For our purposes, a cascade is a data structure in which impressions flow *up* an inverted stair case shaped formation. The stairs in our cascades arise whenever one agent exhaust its budget, at which point the impressions the other agents see flow up from lower slots to higher ones.

Given a ranked list of agents $1, \dots, N$ and the total number of impressions each agent j sees, $T_j \geq 0$, a cascade can be populated as follows. First, agent 1 sees T_1 impressions in the first slot. Second, if $T_2 \leq T_1$, then agent 2 sees T_2 impressions in the second slot; or, if $T_2 > T_1$, then agent 2 sees T_1 impressions in the second slot, and $T_2 - T_1$ impressions in the first slot. In general, the number of impressions that agent j sees in slot j is bounded above by the number of impressions agents 1 through $j - 1$ saw in slot $j - 1$. Similarly, the number of impressions that agent j sees in slot $j - 1$ is bounded above by the number of impressions agents 1 through $j - 1$ saw in slot $j - 2$. And so on.

We formalize this algorithm as a dynamic program in the next section. We then use our formalization to characterize the mathematical properties of cascades.

3.2 Dynamic Program

Given a set of n agents in rank order, a cascade is described by a vector I of length n . The j th entry in this vector is itself a vector of length j . Intuitively, $I_{j,k}$ denotes the number of impressions agent j sees in the k th slot, for $k \in \{j, \dots, 1\}$.

We calculate cascade values (i.e., the entries in “matrix” I) in terms of two auxiliary matrices S and R : $S_{j,k}$

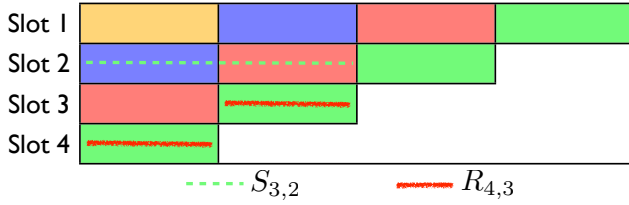


Figure 1: $S_{3,2}$ is the sum of the values of the cells with green dotted lines running through them: i.e., the number of impressions the agents ranked 1 through 3 (the orange, blue, and red agents) see in slot 2. $R_{4,3}$ is the sum of the values of the cells with the red splotchy lines running through them: i.e., the number of impressions agent 4 (the green agent) sees in slots 4 and 3.

records the total number of impressions the agents with ranks 1 through j see in slot k and $R_{j,k}$ records the total number of impressions agent j sees in slots j through k (see Figure 1). Mathematically, $S_{j,k} = \sum_{j'=1}^j I_{j',k}$ and $R_{j,k} = \sum_{k'=k}^j I_{j,k'}$.

Our definition of cascade values is a mutually recursive one, where I is computed in terms of S and R , and vice versa. More specifically, given a ranked list of agents $1, \dots, N$ and the total number of impressions each agent j sees, $T_j \geq 0$, S , R , and ultimately I can be defined inductively as follows:

- for all $j \in \{1, \dots, N\}$ and for all $k \in \{j, \dots, 1\}$,

$$I_{j,k} = \begin{cases} T_j, & \text{if } j = 1 \\ \min\{S_{j-1,k-1} - S_{j-1,k}, T_j - R_{j,k+1}\}, & \text{otherwise} \end{cases} \quad (1)$$

- for all $j \in \{0, \dots, N\}$ and for all $k \in \{j+1, \dots, 1\}$,

$$S_{j,k} = \begin{cases} 0, & \text{if } k = j+1 \\ S_{j-1,k} + I_{j,k}, & \text{otherwise} \end{cases} \quad (2)$$

- for all $j \in \{1, \dots, N\}$ and for all $k \in \{j+1, \dots, 1\}$,

$$R_{j,k} = \begin{cases} 0, & \text{if } k = j+1 \\ R_{j,k+1} + I_{j,k}, & \text{otherwise} \end{cases} \quad (3)$$

These equations form the basis of the dynamic program for computing cascade values presented in Algorithm 1. Initially, agent 1 sees T_1 impressions in slot 1. Subsequently, in the nested for loops, we calculate the number of impressions seen by agents 2 through N in slots $k = j$ through 1. Define R_j as the total number of impressions agent j has seen thus far, and S_k as the total number of impressions agents ranked above j have seen thus far in slot k . If $S_{k-1} - S_k$ (i.e., the difference between the total number of impressions seen thus far by the agents ranked above j in slots $k-1$ and k) is less than the number of impressions agent j has yet to see (i.e., $T_j - R_j$), then a portion of the aforementioned difference is allotted to agent j , but no more than $T_j - R_j$. The quantity S_0 is initialized to ∞ so that $T_j - R_j$ is

always (i.e., for all agents j and for all slots k) less than $S_0 - S_k$. That way, if an agent sees any impressions at all in slot 1, it sees all impressions it has yet to see in that slot.

Algorithm 1 Dynamic Program

Input: a ranked list of agents $1, \dots, N$

Input: the total number of imp'ns T_j , for all agents j

Output: the number of impressions $I_{j,k}$ that each agent j sees in each slot k

$S_0 = \infty$

$I_{1,1} = R_1 = S_1 = T_1$

for $j = 2$ to N **do**

$R_j = 0$

for $k = j$ down to 1 **do**

$S_k = 0$

$I_{j,k} = \min\{S_{k-1} - S_k, T_j - R_j\}$ {the number of impressions agent j sees in slot k }

$S_k = S_k + I_{j,k}$ {the total number of impressions agents see in slot k }

$R_j = R_j + I_{j,k}$ {the total number of impressions agent j sees}

In fact, the inductive definition and dynamic program presented above are only valid when the number of slots M is greater than or equal to the number of agents N . But more commonly, $M < N$. In this more common case, we can proceed as above for all agents with ranks $1, \dots, M$. But to handle the agents ranked below M (i.e., $i \in \{M+1, \dots, N\}$), we create imaginary slots $k \in \{M+1, \dots, N\}$. The number of impressions these agents see in imaginary slots is unconditionally set as follows:

$$I_{j,k} = S_{j-1,k-1} - S_{j-1,k}, \quad \text{if } k \in \{M+1, \dots, N\} \quad (4)$$

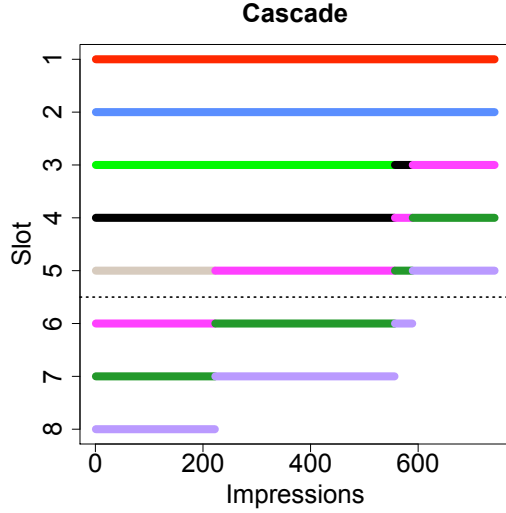
The number of impressions they see in real slots is determined almost as before, except that a comparison is made not simply to $T_j - R_{j,k+1}$, but rather to this same quantity offset by $R_{j,M+1}$, the number imaginary impressions that agent j sees, because the term $R_{j,k+1}$ includes both real and imaginary impressions:

$$I_{j,k} = \min\{S_{j-1,k-1} - S_{j-1,k}, T_j - (R_{j,k+1} - R_{j,M+1})\}, \quad \text{if } k \in \{1, \dots, M\} \quad (5)$$

Figure 2 depicts a sample cascade that was constructed using our dynamic program given the data presented in the corresponding table, assuming 8 agents and 5 slots, as in TAC AA.

3.3 Properties of Cascades

Cascades can be characterized by the following essential properties: 1. For all $j \in \{1, \dots, N\}$ and $k \in \{j, \dots, 1\}$, $I_{j,k} \geq 0$. That is, no agent sees a negative number of impressions in any slot. 2. Impressions run out only once. By this we mean the following: For all $k \in \{M_j, \dots, k^* + 1\}$, $I_{j,k} = S_{j-1,k-1} - S_{j-1,k}$, until some critical slot k^* at which point $I_{j,k^*} = T_j - R_{j,k^*+1}$.



Agent (i)	Impressions (T_i)
1	742
2	742
3	556
4	589
5	222
6	520
7	186
8	153

Figure 2: Sample input to the dynamic program, and corresponding cascade when $N = 8$ and $M = 5$.

Thereafter (i.e., for all $k \in \{k^* - 1, \dots, 1\}$), $I_{j,k} = 0$. Here, $M_j = \min\{j, M\}$.

Equivalently, for all $j \in \{1, \dots, N\}$ and $k \in \{j, \dots, 1\}$,

1. $I_{j,k} \geq 0$.
2. $R_{j,k+1} \leq S_{j-1,k}$.

This property captures the essence of the cascade data structure. In words, the total number of impressions agent j has seen so far (i.e., through slot $k+1$) is always *less than or equal to* the total number of impressions all agents above him in the ranking have seen in slot k .

3. $I_{j,k} > 0 \Rightarrow R_{j,k+1} = S_{j-1,k}$.

In words, if agent j sees any impressions at all in slot k , then it must be the case that the total number of impressions agent j has seen so far (i.e., through slot $k+1$) is *precisely equal to* the total number of impressions all agents above him in the ranking have seen in slot k . It cannot be more, or the aforementioned property would be violated; and it cannot be less, because otherwise there would be an opportunity for agent j to see more impressions in slot $k+1$.

4 Rank and Impression Estimation

En route to solving the rank and impression estimation problem, we first set out to solve only the impression problem (IE). That is, we assume that the agents' ranks are still given, but that total number of impressions seen by each agent other than oneself is not known. Instead, for each agent j , agent i knows only agent j 's average position μ_j , defined as:

$$\mu_j = \frac{\sum_{k=1}^M I_{j,k}k}{T_j} = \frac{\sum_{k=1}^M I_{j,k}k}{\sum_{k=1}^M I_{j,k}} \quad (6)$$

These equations describe N constraints in the IE problem. But recall that agent i also knows the total number of impressions it sees, T_i . This information leads to an $N+1$ th constraint: $T_i = \sum_{k=1}^{M_i} I_{i,k}$. IE is a constraint satisfaction problem (CSP): the entries in the cascade matrix are the variables, and the $N+1$ aforementioned constraints together with the cascade properties comprise the constraints.

Unfortunately, the set of cascades that corresponds to an arbitrary instance of IE is not a singleton. For example, if there are 3 agents, and we, agent 3, see 100 impressions, while the average positions for agents 1, 2, and 3 respectively are 1.0, 2.0, and 3.0, then agent 2 may see any number of impressions greater than 100, and agent 1 may see any number of impressions greater than the number agent 2 sees.

Indeed, the T_j values, for $j \neq i$, are not sufficiently constrained. Hence, we enhance the IE CSP with an objective function that helps select among the various feasible T_j 's. In principle, the objective function is straightforward: an agent should choose the most likely T_j 's given its information, including current observations and past models (i.e., its past predictions of the bids and budgets of the other agents that gave rise to the current cascade). That is, an agent should choose a solution that maximizes $P(T_j' \mid \text{observations, models}) = P(T_1, \dots, T_{i-1}, T_{i+1}, \dots, T_N \mid T_i, \vec{\mu}, \text{models})$, where T_j is a random variable ranging over the total number of impressions agent j sees.

But there are two problems with this approach. The first is that the IE and RIE problems are meant to precede modeling, so it is circular to take models themselves as inputs to these problems. We address this problem in our experimental section, where we vary the models that our algorithms take as input, considering both no models and models based on previous cascade predictions.

The second problem with this approach is that it is unwieldy to calculate the desired probability. We address this problem by simplifying the objective function: first, we (unreasonably) assume independence among all the T_j and then we (reasonably) ignore any dependencies among the T_j 's and the μ_j 's. The probability is then: $\prod_{j \neq i} P(T_j = T_j \mid \text{models})$. Finally, when using historical priors, we assume each random variable T_j is a Gaussian, whose mean value is an exponential moving average $\bar{T}_j$ over the past non-zero total number of impressions predictions for agent j , and whose standard

deviation σ_j is $\bar{T}_j$ multiplied by a large constant, because our priors are not necessarily reliable.

In summary, we define the IE problem as follows: given constants T_i for agent i and μ_j for all agents j , together with variables $I_{j,k}$ for all agents j and slots k and T_j for all agents $j \neq i$,

$$\arg \max_{I_{j,k}, T_j} \prod_{j \neq i} P(T_j = T_j \mid \text{models}) \quad (7)$$

subject to:

$$I_{j,k} \geq 0 \quad \forall j, k \quad (8)$$

$$R_{j,k+1} \leq S_{j-1,k} \quad \forall j, k \quad (9)$$

$$I_{j,k} > 0 \Rightarrow R_{j,k+1} = S_{j-1,k} \quad \forall j, k \quad (10)$$

$$\mu_j = \frac{\sum_{k=1}^{M_j} I_{j,k}}{\sum_{k=1}^{M_j} I_{j,k}} \quad \forall j \quad (11)$$

$$T_j = \sum_{k=1}^{M_j} I_{j,k} \quad \forall j \quad (12)$$

These constraints are exactly the properties described in Sec. 3.3, along with the average position and total impression constraints.

We tackle this problem using two solution techniques: integer linear programming (ILP) and constraint programming (CP). In the former, we simply feed ILPs into a commercial solver; in the latter, we design and implement our own problem-specific tree search.

4.1 Integer Linear Programming

An integer linear program cannot incorporate even the simplified objective function in Equation 7 because it is multiplicative, and hence non-linear. Consequently, we approximate this objective function with one that is additive: $\sum_j \left| \frac{T_j - \bar{T}_j}{\sigma_j} \right|$. This objective function can be represented in a linear fashion as follows:³ We add two constraints per agent: $T_j \leq \bar{T}_j + \sigma_j x_j$ and $T_j \geq \bar{T}_j - \sigma_j x_j$. The objective is then to minimize the total number of standard deviations between T_j and $\bar{T}_j$: $\min \sum_j x_j$.

Our additive, linear objective function approximates the multiplicative one in the following senses: (i) setting T_j close to the mean of the prior is preferred by both objectives; and (ii) if the standard deviation corresponding to a variable T_j is large, then neither objective function is particularly sensitive to the setting of T_j . Figure 3 depicts the two objective functions—the one in Equation 7, and the ILP’s linear approximation.

Aside from the conditional constraints, incorporating the other IE constraints into an ILP is straightforward. Consider the conditional constraint $A \rightarrow B$. This is equivalent to $\neg A \vee B$, and can be broken down into two new constraints, of which only one needs to be satisfied. To allow for only one of these constraints to be satisfied, we introduce a boolean variable, say z , and then add a large constant times z to one of the new constraints,

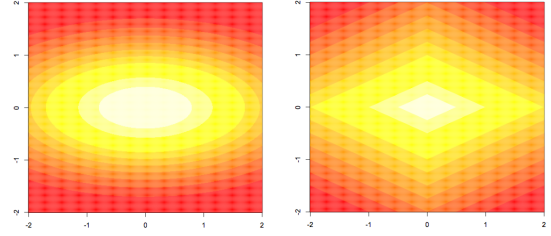


Figure 3: A visualization of the objective function described in Equation 7 for a two-agent cascade (left), and the ILP’s simplified linear objective function (right).

and a large constant times $1 - z$ to the other. Then, depending on whether the boolean variable is true or false, one of the constraints becomes trivial to satisfy, leaving only the other constraint.⁴

In addition to experimenting with this ILP, we also experiment with a LP relaxation of this ILP—a mixed-integer linear program, in which the $I_{j,k}$ (and hence, the T_j) are real- rather than integer-valued.

4.2 Constraint Programming

In addition to using an industrial-strength ILP solver to solve IE (and RIE), we also solve these problems using constraint programming (CP), by which we mean problem-specific tree search. CSPs are often solved using CP solvers (such as Comet [6]). The tree searches these solvers initiate take advantage of the compositionality of well understood constraints and their specialized satisfaction algorithms. Since our applications involve only cascade constraints, which are not particularly well studied, we solve these CSPs using our own light-weight CP solver that is only equipped to solve cascade satisfaction problems.

Like our ILP, our CP solver searches for $I_{j,k}$: i.e., the number of impressions each agent sees in each slot. It begins by assigning agent 1 (equivalently, the agent with average position 1.0) some constant number of impressions, say L , somewhere in the range of 1 and a fixed upper bound T on the maximum total number of impressions (i.e., the maximum number of impressions seen in slot 1). If agent 2’s average position μ_2 is fractional, then μ_2 is necessarily less than 2.0, meaning that agent 2 sees some impressions in slot 1. By the properties of a cascade, agent 2 sees L impressions in slot 2. Given this information, the number of impressions, call it x , that agent 2 sees in slot 1 is fully determined: i.e., $x = \frac{(\mu_2 - 2)L}{(1 - \mu_2)}$, because agent j ’s fractional average position $\mu_2 = \frac{2L + 1x}{L + x}$.

More generally, when the agent with rank j has a fractional average position, the number of impressions it could have seen in slots 2 through j is constrained by the existing cascade: i.e., for all $k = j, \dots, 2$, $I_{j,k} \leq S_{j-1,k-1} - S_{j-1,k} \equiv L_k$. Hence, we fill in impressions

³See http://www.aimms.com/aimms/download/manuals/aimms3om_linearprogrammingtricks.pdf.

⁴Again, see http://www.aimms.com/aimms/download/manuals/aimms3om_integerprogrammingtricks.pdf.

almost exactly as we did in the dynamic program. In the DP, for each agent j (in rank order) and for each slot $k = j \dots 1$, we assign agent j all L_k impressions unless doing so would result in agent j seeing more than T_j impressions in total. This incremental assignment procedure is guaranteed to terminate because the total number of impressions agent j sees is monotonically non-decreasing in the number of impressions seen. In the CP solver, we assign agent j all L_k impressions unless doing so would result in agent j 's average position falling below μ_j . Again, this incremental assignment procedure is guaranteed to terminate because average position is monotonically non-increasing in the impressions agent j sees in higher slots (i.e., slots with lower numbers).

In summary, our CP solvers rely on a subroutine which, for agents with fractional average positions, populates the cascade almost exactly as the dynamic program does. Recall the formula, $I_{j,k} = \min\{S_{j-1,k-1} - S_{j-1,k}, T_j - R_{j,k+1}\}$. The term $T_j - R_{j,k+1}$ represents the number of impressions j has yet to see before reaching its total. Here, $I_{j,k} = \min\{S_{j-1,k-1} - S_{j-1,k}, \bar{I}_{j,k}\}$, where $\bar{I}_{j,k} = (\mu_j \sum_{i=k+1}^j L_i - \sum_{i=k+1}^j i L_i) / (k - \mu_j)$. The term $\bar{I}_{j,k}$ represents the number of impressions agent j would have to see in slot k in order to reach its average position. Hence, agent j sees either $S_{j-1,k-1} - S_{j-1,k}$ impressions in slot k , or if seeing that many would result in an average position that is below μ_j , it sees only $\bar{I}_{j,k}$.

While the number of impressions each agent j with a fractional number of impressions sees in each slot is constrained by the properties of the cascade, the number of impressions agents with whole number average positions see in each slot is unconstrained. Hence, the search within our CP solver is over the number of impressions agents with whole number average positions (e.g., agent 1) see in each slot.

When our CP solver uses historical priors, the search begins at the mean (i.e., the expected number of total impressions), and then zig zags back and forth around the mean with ever-increasing magnitude. When our CP solver operates without historical priors, for agent 1 (with average position 1.0), it searches integers in the range of 1 and T in increments of Δ , the discretization factor; for all other agents j (with whole number average positions greater than 1.0), it initializes its search by assigning j $S_{j-1,j-1}$ impressions in slot j and decrementing from there (again, according to Δ).

Having explained how our CP solver generates cascades (i.e., candidate solutions), it remains to explain how our solver evaluates the quality of these solutions, so that it can return the best one it finds (and prune based on quality). Because we discretize the search space, our CP solver is not guaranteed to satisfy Constraints 11 or 12. Instead, for each candidate solution, we calculate $\hat{\mu}_j = \frac{\sum_{k=1}^{M_j} I_{j,k}}{\sum_{k=1}^{M_j} I_{j,k}}, \forall j$ and $\hat{T}_i = \sum_{k=1}^{M_i} I_{i,k}$. Consequently, the probability of interest here is: $P(\mathcal{T}_1, \dots, \mathcal{T}_{i-1}, \mathcal{T}_i, \mathcal{T}_{i+1}, \dots, \mathcal{T}_N, \mathcal{M}_1, \dots, \mathcal{M}_N, | \mathcal{T}_i, \hat{\mu}, \text{models})$, where $\mathcal{M}_j$ is a random variable rang-

ing over agent j 's average position. As above, we simplify this objective by ignore any dependencies among the $\mathcal{T}_j$'s and the μ_j 's. The new objective function is then: $(\prod_{j \neq i} P(\mathcal{T}_j = T_j | \text{models})) P(\mathcal{T}_i = \hat{T}_i | \mathcal{T}_i) (\prod_j P(\mathcal{M}_j = \hat{\mu}_j | \mu_j))$. Finally, when using historical priors, we assume each random variable $\mathcal{M}_j$ (and $\mathcal{T}_i$) is a Gaussian, with mean μ_j (or T_i) and standard deviation equal to μ_j (or T_i) multiplied by a small constant, because the μ_j 's (and T_i) are reliable statistics.

Our CP solver does not have a time limit; instead time is indirectly controlled by Δ . But in the interest of time, we do prune cascades whenever agent i 's error in average position exceeds that of the best solution found so far.

4.3 Least Discrepancy Search

We solve the rank and impression estimation problem (RIE) in two ways: (1) using an extension of our ILP that searches over rankings as well (which we omit here due to space constraints), and (2) wrapping a least discrepancy (LDS) search around our CP solver, which fixes a ranking of the agents, thereby reducing RIE to IE.

The LDS starts with an initial guess at the ranking. This initial guess is a ranking in which any agent with a whole number average position is ranked in that position, and all other agents are greedily filled into the remaining open positions, with the next lowest available slot being assigned to the remaining agent with the next lowest average position. This initial guess is passed to the CP, which solves the corresponding IE problem. The LDS records the ensuing cascade and its objective value. All two-swaps of this initial ranking are then added to a priority queue, with priority being defined as the sum of the absolute values of the differences in average positions between swapped agents. In this manner, the next guess is the ranking in the priority queue that minimizes the discrepancy between itself and the initial guess. Whenever a ranking is evaluated by the CP, all two-swaps from this ranking are subsequently added to the priority queue (if they have not already been considered). The LDS terminates after evaluating a fixed number of rankings, and the ranking and corresponding cascade with the highest objective value seen is returned.

5 Evaluation

In this section, we evaluate how well our algorithms estimate cascades. We evaluate our algorithms' performance on both the IE sub-problem, where bid ranking is known, and the RIE problem, where bid ranking is unknown.

Our evaluation is done in the context of the TAC AA market game, a simulated ad auction environment that was released as part of the annual Trading Agent Competition (TAC). The TAC Ad Auctions (AA) game [9] captures many of the intricacies faced by advertisers in real ad auctions, such as how to bid on multiple queries and set budgets in the presence of noisy, delayed, and incomplete information. Furthermore, it enables interested parties to benchmark various ad auction designs

and bidding strategies in a quasi-realistic, competitive environment without incurring the monetary risks associated with real-world experimentation.

5.1 Prediction Challenge

To evaluate performance, we created what we call a “prediction challenge” using logs from previously played TAC AA games, similar to the approach taken in other TAC games [14]. The games we run on are a subset of the 2010 TAC AA finals. For each game, auction, day, and agent, if the agent participated in the auction, we look at the aggregate data that the agent received on that day (its own total impressions, and all agents’ average positions), and make predictions about the resulting cascade from that agent’s perspective. Since each agent receives different summary statistics, we can make separate predictions for each auction participant.

As stated earlier, cascade predictions are inputs to other models, such as bid and budget models. But different models rely on different components of the cascade predictions. For example, a model that predicts the total number of searches may only need as input from the cascade the total number of impressions seen by all the agents. On the other hand, models of an opponent’s bids and budgets may depend more heavily on the specifics of the cascade predictions; the time an opponent spent in each slot, for example, should impact predictions about how many clicks that opponent received and how much money it spent. To evaluate the usefulness of our predictions as inputs to various conceivable models, we consider three different forms of cascade predictions: $I_{j,k}$ values are predictions of how many impressions each agent sees in each slot; T_j values are predictions of how many total impressions each agent sees; $T = \sum_j T_{j,1}$ values are the total number of impressions seen during the auction on the given day.

Ground truth $I_{j,k}$, T_j , and T values are determined by manipulating data within the game logs. These logs contain the total number of impressions each agent saw (T_j) as well as the squashed bids of each agent. When each squashed bid is unique, there is a unique bid ordering, and we can infer the $I_{j,k}$ ground truth values using our dynamic programming algorithm. The total number of impressions across agents (T) can then be computed by summing up the impressions seen in the first slot. When each squashed bid is not unique, there can be rare cases where the ground truth values are unknown (for example, if two agents have average position 5.0 and each saw 10 impressions, we cannot determine which of these agents was shown first). We throw out such instances. Of the 9280 auctions that occurred in our data set of 10 games, 16 queries, and 58 days, we threw out 83 instances due to ambiguous squashed bid rankings.

For each type of impressions prediction we make, we look at the ground truth and compare it to the predicted value. We report the mean absolute difference between these predicted and actual $I_{j,k}$, T_j , and T values. Absolute instead of relative error is reported because ground truth impression per slot values can be zero, leading to

undefined relative error, or they can be very small (e.g., when agents place probing bids), leading to artificially high relative error values. For ranking predictions, we report the fraction of instances in which the correct ranking is predicted for the RIE problem. All predictions are evaluated with respect to time; the CP is given different discretization factor (Δ) values which determine time spent, while the ILP and MIP are given a maximum amount of time that can be spent on a given instance.

Because agents are playing a repeated game and receive summary statistics at the start of each time period, they may be able to use their past cascade predictions to inform their current ones. To test this effect, in some experiments, we endow agents with historical priors: i.e., prior beliefs about each opponent’s total number of impressions based on past predictions. In other experiments, we assume a uniform prior, which amounts to assuming agents have no prior beliefs about their opponents’ impressions.

5.2 Algorithmic Performance

Figure 4 shows the results of the prediction challenge when run on both the IE and RIE problems from the point of view of our own agent, Schlemazl [4]. For all three metrics of impression prediction accuracy, the CP is the most accurate predictor for smaller time limits, until a certain threshold where the ILP becomes more accurate than the CP. The MIP is faster than the ILP but never more accurate than the CP, which suggests that integer decision variables are important for ILP accuracy. For TAC AA’s specific time limits, the CP makes the most accurate impression predictions.

Among the predictions made for $I_{j,k}$, T_j , and T , error is lowest when predicting impressions per slot. This is a combination of two effects: first, $I_{j,k}$ values are smaller, so there is less room for error; second, $I_{j,k}$ predictions are often correct except for the number of impressions seen by the top agent in the first slot. When an agent in the first slot sees more impressions than anybody else, that agent could have seen an arbitrarily high number of impressions without affecting the impression per slot predictions for the other agents.

For all algorithms and error metrics, impression error is lower in the IE problem than the RIE problem. This is unsurprising, because incorrect ranking predictions in the RIE problem will lead to greater impression error. Figure 5 (Left) shows the fraction of instances of the RIE problem for which the correct ranking was predicted. Ranking accuracy improves with time, and like the impression accuracy results, the CP outperforms the ILP at smaller time limits. As the time limit increases, the LDS wrapped around the CP does not check more possible rankings, but its objective value for each ranking becomes more accurate, since its discretization factor for the resulting IE problem is smaller at larger time limits. We also see that the poor performance of the MIP can be explained by its incorrect ranking predictions. Allowing real-valued impression variables appears to greatly increase the space of feasible rankings, and the

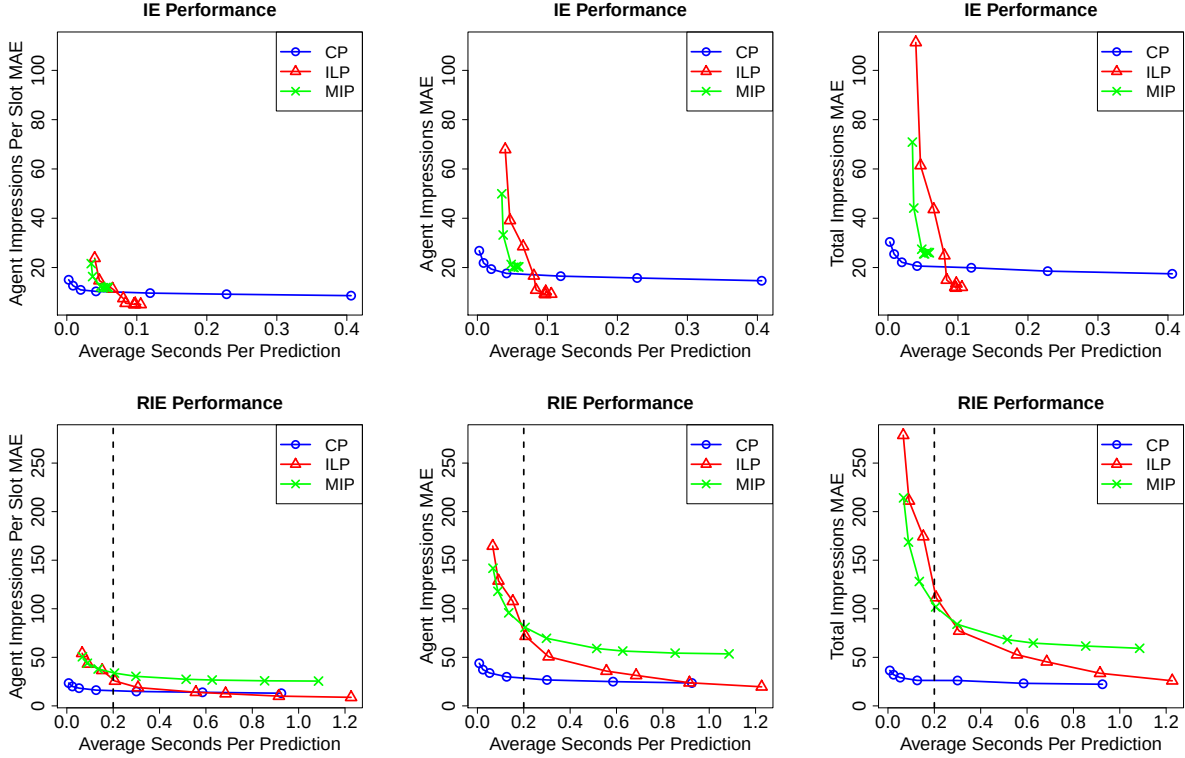


Figure 4: Accuracy of impression predictions for the IE (top row) and RIE (bottom row) problems. The left column shows predictions for agent impressions per slot, the middle column for agent impressions, and the right column for total impressions across agents. The black dashed line is approximately the amount of time an agent would have to build each of its per-query cascade predictions in TAC AA, to allow time for other predictions and optimization.

MIP assigns some of these rankings high objective values, but with integer-valued impression variables, these rankings are not considered.

Using historical priors led to mixed results, improving the CP’s accuracy but not the ILP’s or the MIP’s. Historical priors are not based on ground truth but on past cascade predictions, so there is potential for a positive feedback loop which can cause predictions to become increasingly inaccurate. Similarly, accurate predictions will tend to reinforce themselves. Figure 5 (Right) illustrates how the error in predictions with historical priors compares favorably to the error in predictions with uniform priors for the CP. Historical priors are perhaps taking advantage of situations in which the day-to-day IE and RIE problems are underconstrained only sporadically. When a majority of consecutive instances have unique solutions, those solutions can be used to bootstrap predictions on more difficult instances: i.e., when many agents have whole number average positions.

5.3 Agent Behavior

We now examine prediction performance differences of individual agents for just one of the problems (RIE, uniform priors) and one of our algorithms (CP) (see Table 1). Surprisingly, some of our worst predictions occur

when we take on the role of Mertacor and TacTex, two of the top three finishers in the 2010 TAC AA tournament. On the other hand, Mertacor does make the best total impressions predictions. Tau achieves relatively accurate predictions on all dimensions. Further analyzing individual agent behavior can lead to some insights about why performance varies across agents.

TacTex appears in more auctions than any other agent, and often receives a whole number average position, but is rarely in the first slot. Additionally, TacTex often drops out of the auction early, observing only a small amount of the day’s total impressions. This means that TacTex is receiving daily summaries that are not particularly helpful for predicting total impressions—the agent’s number of impressions typically constrains the space of feasible T values, but receiving only a few impressions does not provide a very tight lower bound. Additionally, agents that are in the auction for a longer period of time are more likely to have fractional average positions. A fractional average position together with a large total number of impressions often substantially limits the number of impressions that agent could have seen in each slot, which in turn constrains how many impressions the other agents could have seen in each slot.

However, it is worth noting that TacTex is at a disad-

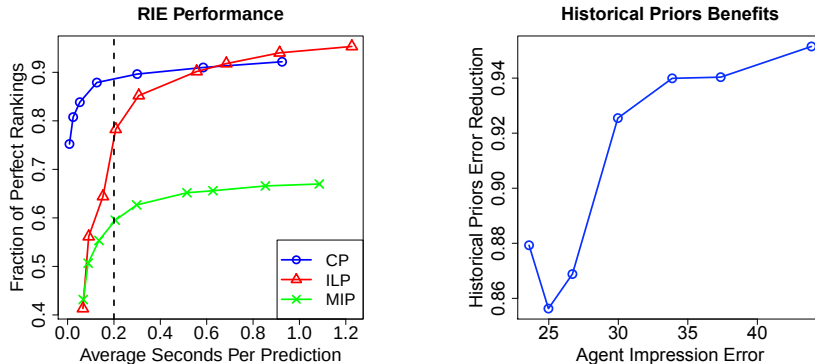


Figure 5: Left: Fraction of times the algorithms produce the correct rankings in the RIE problem. Right: Benefit from using historical priors with the CP algorithm. Due to the inherent feedback loop in historical priors, as agent impression error (with uniform priors) improves, historical priors become more effective.

Agent	Performance				Behavior				
	$\varepsilon(I_{j,k})$	$\varepsilon(T_j)$	$\varepsilon(T)$	rank	N	%F2	%int	%top	%seen
TacTex	18	39	60	.94	33686	69	71	12	26
Schlemazl	16	30	26	.88	14821	91	16	9	72
Mertacor	29	54	9	.83	8449	80	12	7	98
MetroClick	20	34	35	.89	14363	88	15	6	77
Nanda_AA	15	27	10	.89	12675	80	13	8	90
crocodileagent	17	27	29	.92	25451	89	17	9	67
tau	10	19	11	.92	19205	79	94	91	32
McCon	19	32	47	.95	29810	68	22	14	58

Table 1: Agent-specific prediction accuracy for the RIE problem with uniform priors using the CP algorithm. The best prediction of each type is shaded a light gray, while the worst prediction is shaded a dark gray. Descriptions of behavior columns, from left to right: number of predictions made, percentage of predictions made in the F2 focus level (a particular type of highly focused query in TAC AA), percentage of auctions where the agent had an integer average position, percentage of auctions where the agent was in the first slot, average percentage of impressions seen before the agent dropped out.

vantage in our prediction challenge because in it, agents are only required to make predictions when they actually participate in an auction, and TacTex participates in the most auctions. Seeing even a few impressions probably leads to better cascade predictions than seeing no impressions at all. Additionally, by dropping out of the auction early, TacTex causes other agents to have fractional average positions. While this improves cascade predictions overall, other agents are better able to capitalize because their own total impressions values are more informative. Even if TacTex’s cascade predictions suffer when it drops out, its other models are likely improved because it sees cost information on a daily basis in a number of different queries.

Mertacor’s strategy leads to notably different behavior than TacTex, as it appears in the smallest number of auctions. However, when it does appear in these auctions, it is frequently in them for the duration. This allows it to have nearly-perfect T predictions.

Tau never appears in F0 queries (not shown), which are the most general queries with the highest number of searches per day, and thus have the most potential for

error. Additionally, we see that tau is very frequently in the first slot. The number of impressions seen by the agent in the first slot is often underconstrained, but tau does not run into problems making this prediction because its usually occupies this position itself.

The notable difference in performance of agents’ cascade predictions brings up the issue of the coupled nature of predictions and decisions; specifically, the ability to make decisions not just to maximize profit given current predictions, but to improve predictions for future days. While some agents do take ad-hoc approaches to probing, explicitly calculating the value of information, as far as we know, is not typical of agents in TAC AA or the other TAC games. Our results suggest that such calculations could be important for improving agent performance. We leave this as a direction for future work.

6 Related Work

The problem of disaggregating average position data to predict advertiser rankings and impressions within the context of TAC AA was first studied by the TacTex team [13]. Their approach to the RIE problem utilized

a tree-based search over three values for each advertiser: squashed bid rank, the order that spending limits were reached, and a GCD multiplier. The TacTex algorithm reduces the search space dramatically by searching over GCD multipliers and takes advantage of heuristics to further prune the space. Our paper is a natural extension of their work: we formalize the mathematical properties of cascades in a stylized model of GSPs and present an ILP based on these properties. We also present a CP solver that runs faster than our ILP. An additional contribution of our paper is a closer investigation of how an agent's average position and total number of impressions seen affect the accuracy of its cascade.

7 Conclusion

Recent work by Abhishek *et al.* [1] shows both analytically and empirically, with actual search engine data, that building models from average position data can lead to aggregation bias; for example, predicted click probabilities can exceed their actual values. Our work directly addresses the problems identified by these researchers, and could potentially be used to mitigate such bias.

In this paper, we develop algorithms for disaggregating ad auction data within a stylized version of the GSP: i.e., for unraveling an advertiser's daily average position into its actual positions throughout the day. We began by formulating and solving via dynamic programming a complete information version of the disaggregation problem. We then formalized the mathematical properties of cascades (i.e., solutions to this problem), and used the cascade properties to guide our design of a constraint programming algorithm and a mathematical programming model.

We show that (1) the ILP and CP are both well-suited to solving the rank and estimation problem, and the choice of which to use should depend on time constraints; (2) using historical priors on opponent impressions improved accuracy for the CP, despite a potential feedback loop; and (3) prediction accuracy differs depending on the agent's bidding behavior. This third point shows the potential importance of making decisions not just to maximize profit given current predictions, but also to improve predictions so that the inputs to the optimization routine are more accurate on subsequent days. As far as we know, no TAC AA agents make any explicit value of information calculations, though they might probe for information in an ad-hoc manner; our results suggest that this may be a fruitful direction for future research. Additional future research will investigate how the accuracy of our cascade predictions affect the accuracy of other predictions such as opponent bids and budgets.

TAC AA assumes a stylized model of generalized second price auctions, with more straightforward slot dynamics than most real-world ad auctions. For example, in TAC AA, the ads of top ranking advertisers are consistently displayed until those advertisers' budgets are exhausted; but some prominent Internet publishers do not show all the top ranking advertisers, because they

attempt to control the rates at which advertisers spend their budgets. Hence, our algorithms would need to be extended before they could be used by advertisers in some important real-world ad auctions. Nonetheless, we believe that the present work is a necessary first step towards solving any real-world analogue of RIE, and hence potentially mitigating the aggregation bias reported by Abhishek *et al.* [1].

8 Acknowledgments

This research was supported by NSF Grant CCF-0905234 and the Centrum Wiskunde and Informatica (CWI).

References

- [1] V. Abhishek, K. Hosanagar, and P.S. Fader. Identifying the Aggregation Bias in Sponsored Search Data. 2011.
- [2] Gagan Aggarwal, Ashish Goel, and Rajeev Motwani. Truthful auctions for pricing search keywords. In *Proc. ACM EC*, pages 1–7, June 2006.
- [3] S. Athey and G. Ellison. Position auctions with consumer search. *Working Paper*, 2007.
- [4] Jordan Berg, Amy Greenwald, Victor Naroditskiy, and Eric Sodomka. A first approach to bidding in ad auctions. In *TADA '10: Workshop on Trading Agent Design and Analysis*, June 2010.
- [5] Christian Borgs, Jennifer T. Chayes, Nicole Immorlica, Mohammad Mahdian, and Amin Saberi. Multi-unit auctions with budget-constrained bidders. In *Proc. ACM EC*, pages 44–51, 2005.
- [6] Dynadec, Inc. Comet 2.1 User Manual. <http://dynadec.com/>, 2009.
- [7] Benjamin Edelman, Michael Ostrovsky, and Michael Schwarz. Internet advertising and the generalized second price auction: Selling billions of dollars worth of keywords. NBER Working Paper No. 11765, 2005.
- [8] A. Greenwald, V. Naroditskiy, and S. J. Lee. Roxybot-06: Stochastic prediction and optimization in TAC travel. *Artificial Intelligence Research*, 36:513–546, 2009.
- [9] P.R. Jordan and M.P. Wellman. Designing an ad auctions game for the trading agent competition. In *Workshop on Trading Agent Design and Analysis*, July 2009.
- [10] Brendan Kitts and Benjamin J. LeBlanc. Optimal bidding on keyword auctions. *Electronic Markets*, 14(3):186–201, 2004.
- [11] Sébastien Lahaie and David M. Pennock. Revenue analysis of a family of ranking rules for keyword auctions. In *Proceedings of the 8th ACM conference on Electronic commerce*, EC '07, pages 50–56, New York, NY, USA, 2007. ACM.
- [12] Aranyak Mehta, Amin Saberi, Umesh V. Vazirani, and Vijay V. Vazirani. Adwords and generalized on-line matching. In *Proc. FOCS*, pages 264–273, 2005.
- [13] D. Pardoe, D. Chakraborty, and P. Stone. TacTex09: A champion bidding agent for ad auctions. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, 2010.
- [14] David Pardoe and Peter Stone. The 2007 tac scm prediction challenge. In *AAAI 2008 Workshop on Trading Agent Design and Analysis*, 2008.
- [15] Hal R. Varian. Position auctions. *International Journal of Industrial Organization*, 25(6):1163–1178, December 2007.
- [16] Michael P. Wellman, Amy Greenwald, and Peter Stone. *Autonomous Bidding Agents: Strategies and Lessons from the Trading Agent Competition*. MIT Press, 2007.

An Adaptive Proportional Value-per-Click Agent for Bidding in Ad Auctions

Kyriakos C. Chatzidimitriou
Aristotle University of Thessaloniki
Centre for Research and Technology Hellas

Lampros C. Stavrogiannis
University of Southampton, UK

Andreas L. Symeonidis and **Pericles A. Mitkas**
Aristotle University of Thessaloniki and Centre for Research and Technology Hellas
mertacor@olympus.ee.auth.gr

Abstract

Sponsored search auctions constitutes the most important source of revenue for search engine companies, offering new opportunities for advertisers. The Trading Agent Competition (TAC) Ad Auctions tournament is one of the first attempts to study the competition among advertisers for their placement in sponsored positions along with organic search engine results. In this paper, we describe agent Mertacor, a simulation-based game theoretic agent coupled with on-line learning techniques to optimize its behavior that successfully competed in the 2010 tournament. In addition, we evaluate different facets of our agent to draw conclusions about certain aspects of its strategy.

1 Introduction

The advent of the Internet has radically altered current business landscape. One of its prominent applications is on-line advertising in search engine results, known as sponsored search, where paid advertisements are shown along with regular results (called impressions). Sponsored search is the highest source of revenue for on-line advertisement, yielding a profit of approximately \$10.67 billions for 2009 only in the U.S. [PwC, 2010].

In the sponsored search setting, whenever a user enters a query in the search engine (publisher), an auction is run among interested advertisers, who must select the amount of their bids, as well as the advertisements that they deem to be more relevant to the user. There is a number of positions (slots) for placement, but higher slots are more desirable, given that they generally yield higher levels of Click-Through-Rate (CTR). This field started in 1998 by GoTo.com, where slots were allocated via a Generalized First Price (GFP) auction, but received its current form in 2002, when GFP was replaced by the well known Generalized Second Price (GSP) auction [Jansen and Mullen, 2008]. According to this auction, bids are sorted by bid (that is usually multiplied by an advertiser-specific quality factor), and the winner of a slot pays the minimum bid needed to get this position, which is slightly higher than the next bidder's offer and independent of her bid. What makes this type of auctions different is the fact that payment is made on a per click (cost-per-click

or CPC) rather than a per impression (cost-per-mille or CPM) basis.

Against this background, we present the strategy of agent *Mertacor*, our entrant that participated in the TAC Ad Auctions 2010 competition [Jordan *et al.*, 2010] and was placed 3rd in the finals. At a high level, Mertacor's strategy can be decomposed into two parts: (a) estimating the Value-per-Click (VPC) for each query and (b) choosing a proportion of VPC for bidding in each auction based on the state of the game (the adaptive proportional part). The approach is similar to the QuakTAC agent [Vorobeychik, 2011], which participated in the 2009 competition, with two extensions: (a) a k-nearest-neighbors algorithm to help in the estimation of VPC and (b) an associative to the state of the game, n-armed bandit formulation of the problem of selecting the proportion of VPC to bid.

The remainder of this paper is organized as follows: Section 2 provides a brief description of the game. Section 3 presents strategies of agents participated in the previous competition. Section 4 builds the background upon which our agent was based and gives a detailed description of the extension points. An discussion of the conducted experiments is given in Section 5. Finally, Section 6 concludes the paper and provides our future research directions.

2 The TAC Ad Auctions Game

Sponsored search auctions are open, highly complex mechanisms, that are non-dominant-strategy-solvable, hence bidding strategies are a topic of active research. To investigate their behavior, a realistic agent-based simulator seems essential [Feldman and Muthukrishnan, 2008]. The Ad Auctions (AA) platform in the international Trading Agent Competition (TAC) is such a system. The TAC AA game specifications are defined in detail in [Jordan *et al.*, 2010]. To familiarize the reader with the game, we will provide some basic information about the entities involved and the interactions between them.

In TAC AA tournament, there are three main types of entities, the publisher, a population of 90000 users, and eight advertiser entrants represented by autonomous software agents. The advertisers compete against each other for advertisement (ad) placement, across search pages. Each one of the search pages contains search engine results for one

of the queries of 16 different keyword sets. In order to promote their products, the agents participate in ad auctions by submitting a bid and an ad to the publisher for the query (set of keywords) they are interested in. Ads are ranked on each search page, based on a generalized method that interpolates between rank-by-bid and rank-by-revenue schemes. Each day, users, according to their preferences and state, remain idle, search, click on ads and make purchases (conversions) from the advertisers' websites. The products being traded are combinations of three brands and three types of components from the domain of home entertainment. The small number of products enables competing teams to focus only on a small set of predefined keywords, abstracting away from the problems of keyword selection. The three manufacturers (namely, Lioneer, PG and Flat) and the three types of devices (TV, Audio and DVD) constitute a total of nine products. The simulation runs over 60 virtual days, with each day lasting 10 seconds. A schematic of the interactions between game entities is found in Figure 1.

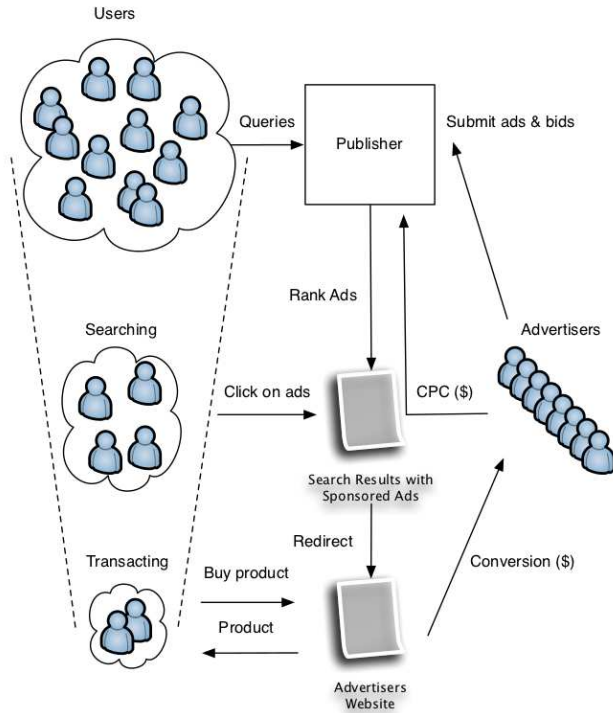


Figure 1: The entities participating in a TAC AA game along with their actions. The majority of the users submit queries, fewer click on ads and an even smaller percentage makes transactions.

2.1 Advertisers

Each advertiser is a retailer of home entertainment products and can supply the user with any of the nine products available. Upon initialization of each simulation, advertisers are given a component and a manufacturer specialty, yielding an increase in conversion rates for the former and

an increase in profit per unit sold for the latter. Additionally, entrants are assigned a weekly maximum stock capacity $C^{cap} \in \{C^{LOW}, C^{MED}, C^{HIGH}\}$, so conversions above this threshold are less likely to happen during this week (5 working days). During its daily cycle of activities the advertiser agent on day d has to:

- Send the bids for ad placement per query for day $d + 1$.
- Select an ad for ad placement per query for day $d + 1$. The ad can be either generic (i.e. referring to a general electronics shop) or targeted (i.e. stating a specific manufacturer-component combination). A targeted ad that is a match to the user preferences increases the probability of clicking that ad.
- Set spending limits for each query and across all queries for day $d + 1$.
- Receive reports about the market and its campaign for day $d - 1$.

2.2 Publishers

As mentioned above, the publisher runs a GSP auction to determine the rank of bids and determine the payment per click. The ad placement algorithm takes into account predefined reserve scores. There is one reserve score below which an ad will not be posted, and one above which, an ad will be promoted. If the spending limit set by an agent is passed, the rankings are recalculated. The auction implemented is a GSP, where the ranking takes into account the quality of the advertisements, weighted by a squashing parameter that is disclosed to the entrants at the beginning of the game.

2.3 Users

Each user has a unique product preference and can be in different states representing his or her searching and buying behavior (i.e. non-searching, informational searching, shopping, with distinct levels of shopping focus, and transacted). The product preference distribution is even for all products. Users submit three kinds of queries, defined by their focus level for a total of 16 queries. There is one (1) $F0$ query, where no manufacturer or component preference is revealed, six (6) $F1$ queries, where only the manufacturer or the product type is included in the query and nine (9) $F2$ queries, where the full product definition (manufacturer and type) is exposed. Users' daily state transition is modeled as a Markov chain. Non-searching and transacted agents do not submit queries. Informational agents submit one of the three queries by selecting any one of the them uniformly and focused users submit a query depending on their focus level. While both information seeking and focused users could click on an ad, only focused users make purchases and go to the transacted state. Users click on the advertisements based on an extended version of the cascade model [Das *et al.*, 2008]. After clicking on an ad, whether a conversion will be made or not depends on user's state, advertisers' specialty and remaining distribution capacity.

3 Related Work

Given that the tournament started two years ago, relevant published work on TAC AA is limited. The majority of strategies

is focused on two target metrics, namely the *Return on Investment* (ROI), i.e. the ratio of profit to cost, and the *Value per Click* (VPC), i.e. the expected profit from a conversion given a click, and combined with multiple choice knapsack (MCKP) models to deal with the distribution constraint effect.

TacTex [Pardoe *et al.*, 2010], the winner in the previous competitions, implements a two-stage strategy of estimation and optimization. The former incorporates self and opponent related predictions of desired variables as well as user state estimation. More specifically, this agent tries to extract the range of impressions, ranking and amount of bids, as well as the type of ads shown. Then, it estimates the proportion of users in each state and utilizes this information to predict the most profitable number of conversions per query type. The optimization takes into consideration the distribution constraint effect, hence both short-term and long-term optimizers are incorporated in the strategy. Finally, there is an estimator for the unknown game parameters, i.e. continuation probability and quality factor.

Another competitor for 2009, *Schlemazl*, employs rule-based as well as dynamic programming techniques. According to the latter, the bidding process is modeled as a penalized MCKP, where the value of each query is the profit made and the weight is the number of sales [Berg *et al.*, 2010]. For the rule-based method, the agent’s strategy targets for the same ROI in all queries. A similar principle is implemented in *EPFLAgent 2010* [Cigler and Mirza, 2010], targeting for the same profit per conversion over all queries, and distributes the number of sales uniformly on the five most recent days. If the number of sales exceeds this threshold, the bid for the most profitable keyword is increased, otherwise the bid on the least profitable keyword is decreased.

Tau [Mansour and Schain, 2010] follows a reinforcement learning (soft greedy) approach, based on regret minimization, where the agent selects a value from a distribution on the space of bids, so that the regret, which is the difference between actual profit and maximum profit, is minimized. On the other hand, *DNAgent* [Munsey *et al.*, 2010] follows an evolutionary computation technique, by creating a population of agents (referred as finite state machines) that are genetically evolved until the fittest is determined. Each agent can be in seven different states, based on its previous position, which is combined with the matching of the advertisement to the user to determine bid adjustments for the next day.

Finally, *AstonTAC* [Chang *et al.*, 2010] and *QuakTAC* [Vorobeychik, 2011] follow VPC-based strategies. The former estimates the market VPC, which is the VPC minus relevant cost, and then bids a proportion of this value based on the critical capacity (i.e. capacity beyond which the expected cost is higher than the corresponding revenue) and the estimated quality factor for the advertiser. Priority is also given to the most profitable queries. On the other hand, *QuakTAC* follows a simulation based Bayes-Nash equilibrium estimation technique to find the optimum percentage of the VPC to bid. One important advantage of the VPC strategy is that it does not require any opponent modeling techniques, which in turn, demand historical information about opponents’ bidding behavior, difficult to obtain in reality.

4 Agent Mertacor

4.1 Background

The baseline strategy of agent Mertacor is a modified version of the aforementioned QuakTAC strategy for the 2009 tournament [Vorobeychik, 2011]. This is one of the few reported strategies in TAC that employs simulation based game theoretic analysis and was proven quite successful in that tournament, as QuakTAC was placed 4th in the finals. It is a sound, elegant and yet simple strategy. For the bidding part, Vorobeychik considers a simple strategy space, with bids that are linear to the valuation of the player. For the AA scenario, this valuation is the advertiser’s VPC, v , and the agent bids a proportion (shading), α , of this value. For each query q , the bid for day $d + 1$ would be:

$$bid_{d+1}^q = \alpha \cdot v_{d+1}^q$$

Given that GSP auctions are not incentive compatible [Lahaie, 2006], we expect that $\alpha < 1$. An estimation of the VPC for a keyword can be expressed as the product of the probability of converting after a click on the ad and the expected revenue from such a conversion. So for each query q , the agent’s estimated VPC value can be calculated as:

$$\hat{v}^q = \hat{Pr}^q\{conversion|click\} \cdot E[revenue^q|conversion]$$

The expected revenue for a query q given a conversion, ($E[revenue^q|conversion]$), solely depends on the advertiser’s Manufacturer Specialty (MS) and can be calculated with no additional information for the three possible cases as follows:

$$\begin{cases} (USP \cdot (3 + MSB))/3 & \text{MS not defined in } q \\ USP \cdot (1 + MSB) & \text{MS matched in } q \\ USP & \text{MS not matched in } q \end{cases}$$

where USP is the unit sales profit (\$10 in TAC 2010) and MSB the manufacturer specialty bonus (0.4 in TAC 2010).

However, for calculating the probability of conversion, we need two additional estimations, the proportion of focused users and past and future conversions that have an impact on the conversion probability due to the distribution constraint effect:

$$\hat{Pr}_{d+1}^q\{conversion|click\} = focusedPercentage^q \cdot Pr^q\{conversion|focused\}(\hat{I}_{d+1})$$

To calculate a value for the $focusedPercentage^q$ estimate, we used the following procedure. If we fix advertisers’ policies, then the proportion of focused users is equal to the ratio of the clicks that result in conversions divided by their individual probability of conversion (incorporating the distribution constraint effect). The user population model in TAC AA is a Markov chain and, despite the incorporation of the burst transitions, it will typically converge to a stationary distribution. Hence, instead of using a more sophisticated and accurate particle filter, we have used Monte Carlo simulations, running 100 games with symmetric, fixed strategy profiles, making sure that their bids are always higher than those requested by reserve scores, and have then recorded the

mean values for our desired ratios (every day results for individual queries are aggregated for each focus level). Historical proportions are kept in log files and are updated incrementally. In our experiments, we have used last day's probability of conversion as a close estimate of the requested value for the denominator.

Our first differentiation compared to the original strategy lies in the estimation of the conversion probability, $Pr^q\{conversion|focused\}(\hat{I}_{d+1})$, which is a function of the distribution constraint effect, I_d . As Jordan has noticed in his thesis [Jordan, 2010], the distribution constraint effect, I_d , is the second most influential factor in an advertiser's performance after the manufacturer specialty in terms of the coefficient of determination values. This effect radically affects the probability of purchase. Based on the specifications of the game for the calculation of the distribution constraint effect for day $d+1$, we need, but don't have the conversions, c , of day d and $d+1$:

$$I_{d+1} = g(c_{d-3} + c_{d-2} + c_{d-1} + \hat{c}_d + \hat{c}_{d+1} - C^{cap})$$

where g is a function defined in the specification that gives the value of I_d .

Given that an entrant must calculate a bid with a 2-day lag in available information, QuakTAC estimates tomorrow's I_{d+1} value using a two-step bootstrapping technique. More specifically, it estimates conversions on day d as the product of current game's historical click-through rate, recorded historical impressions and (optimistically) estimated conversion probability. Then it uses this information to estimate day's $d+1$ conversion rate and corresponding conversions and uses an averaged value for the I_{d+1} .

Having implemented that strategy, we realized that it performs poorly on the 2010 server specifications. It seems that it underestimates the VPC, systematically bidding much lower than what is allowed by the publisher's reserve scores. It is important to note that these scores for 2010 are much higher than 2009. Jordan describes the effect of the reserve score on publisher's revenue as well as players' equilibrium showing that AstonTAC and Schlemazl perform much better than TacTex for these higher reserve scores. To rectify this shortcoming, we have implemented a simpler but more reasonable expectation: we aggregate last three days' conversions and set $\frac{1}{4}$ of this value as our number of conversions for day d as well as day $d+1$. This is slightly lower than their corresponding mean value so as to intentionally be more conservative. However, when our capacity is C^{LOW} , we have used instead the mean value, which was experimentally shown to yield better results in this case.

Finally, we had to select the percentage of our VPC to bid. Following the methodology of Vorobeychik, we restrict our interest in simple symmetric equilibria and discrete values ranging from 0 to 1 with a step of 0.1. This means that all but one advertisers follow the same strategy ($bid = \alpha v$) and a single player bids another α_{single} percentage, which varies among games. The values for α_{single} are determined via an iterative best response method, where we start from a truthful homogeneous strategy profile ($\alpha = 1, \alpha_{single} = 1$). This is a reasonable value to start, as GSP is considered a generalization of the Vickrey auction. Then we find the best

α_{single} deviation from this profile and use this last value as the new homogeneous profile. This process is repeated until self-response is a Bayes-Nash equilibrium. We have validated the speed of this method, being able to get the best value of $\alpha_{single} = 0.3$ in three iterations (1, 0.4 and 0.3). The alpha value of 0.3 and the new method for calculating I_d are the differentiation we made to the agent playing in the qualifications round of TAC AA 2010 and further defined as *Mertacor-Quals*.

Ad Selection Strategy

It is also important to describe our ad selection strategy. This task is straightforward for $F0$ (no keywords) and $F2$ (containing both brand and product keywords) type queries. For the first case, there is a $\frac{1}{9}$ probability of matching a user's preference, so a generic ad seems most appropriate. On the other hand, users reveal truthfully their preferences, hence a targeted $F2$ ad will always match them. For the $F1$ keywords (containing either brand or product keywords), we have also used a targeted advertisement, where our agent's respective specialty is shown when the manufacturer or the component are not disclosed, hoping that our increased profit or conversion probability benefits will outweigh our higher probabilities of mismatch. This strategy also proved to be effective in the original QuakTAC agent [Vorobeychik, 2011].

4.2 Extensions

For the final rounds, we tried to improve our agent in the two critical calculations of: a) estimating the distribution constraint effect and b) finding an appropriate α value by adapting it on-line according to the current market conditions.

In the first case, we used the k-Nearest-Neighbors (k-NN) algorithm to estimate the capacity to be committed for the current day c_d and then using that estimate to further predict the capacity to be purchased on day c_{d+1} . We chose $k = 5$ and stored 5-day tuples for the last 10 games (600 samples). We adopted this kind of modeling because the agent executes a clear 5-day cycle in its bidding and conversion behavior. This behavior is derived from the strategy of the agent since when having enough remaining capacity, it will produce high bids due to high VPC values and get ranked in the first positions. That way it will receive many clicks and a high volume of conversions will take place. This behavior will continue for one more day, until the agent's store is depleted. Then the bids will be low and the agent will maintain a no conversions status for three days.

In order to choose the proportion of the VPC to bid in a real antagonistic environment, we have formulated the problem of choosing α into an associative n -armed bandit problem [Sutton and Barto, 1998]. Under this formulation, we switched bidding from $b(v) = \alpha \cdot v$ to $b(\alpha, v) = \alpha \cdot v$.

Based on the experience gained from the TAC SCM domain, we tried to maintain a low dimensionality in the state and action spaces for sampling efficiency. We chose VPC and query types as state variables. The choice of VPC was made because its calculation incorporates many parameters that characterize the state of the agent and the market such as: specialties, distribution capacity constraint, and proportion of focused users. The query type was added as state variable in

order to provide additional information about the current state from another dimension. VPC was quantized over 11 states, 10 for VPC values between \$0 and \$10 and 1 for VPC values above \$10. The 16 query types were mapped into 8 states as presented in Table 1. There were 6 discrete actions picking values of α between equal spaced points from 0.275 to 0.325 inclusive. So a total of 528 $Q(s,a)$ values need to be stored and updated. For exploration purposes ϵ -greedy policy was used with $\epsilon = 0.1$. The goal is to learn a policy for selecting one of the 6 actions that would maximize the daily profit given the state of the game. Updates to Q values were made using the learning rule:

$$Q(s, a)_{k+1} = Q(s, a)_k + 0.1 \cdot (r_k - Q(s, a)_k)$$

Reward is the daily profit calculate as the revenue minus the cost for the corresponding state and action.

$$r = \text{revenue} - CPC \cdot \#clicks$$

On day d we receive the report for day $d - 1$ from which we can calculate the reward for day $d - 1$. That reward concerns the bids made on day $d - 2$ for day $d - 1$.

Table 1: Mapping query types to states. One state for the $F0$ query, 3 states for $F1$ queries and 4 states for $F2$ queries.

Query Type	State ID
F0	1
F1 with no matching specialty	2
F1 with matching manufacturer specialty	3
F1 with matching component specialty	4
F2 with no matching specialty	5
F2 with matching manufacturer specialty	6
F2 with matching component specialty	7
F2 with all matching specialties	8

5 Analysis

To validate the effectiveness of this iterative best response technique, after the tournament we have also repeated the experiments for all possible combinations of homogeneous and single α values and have plotted the best α_{single} as a function of the symmetric opponent profile. Each experiment was repeated 5 times for a total of 500 log files. Results are given in Figure 2. As can be seen, there are only two close values for $\alpha_{single}(0.3, 0.4)$, although 0.3 is best response for more reasonable strategies ($\alpha < 0.5$). Hence, this technique yields robust results in only 3% of total required time for extensive experiments. Moreover, we should note that the values of 0.1 and 0.2 used by QuakTAC last year are not profitable at all in this year's platform due to the aforementioned reserve score effect.

The final version of agent Mertacor got the third place in the TAC AA 2010 competition. The standings are shown in Table 2.

To evaluate the effectiveness of the agent and the extensions incorporated, two tournaments were conducted. Four versions of agent Mertacor were constructed. *Mertacor-Quals* is the agent that participated in the qualifications of

Table 2: TAC AA 2010 tournament results.

Position	Agent	Average Score (\$)
1	TacTex	58 130
2	Schlemazl	52 868
3	Mertacor	52 383
4	MetroClick	52 210
5	Nanda_AA	48 084
6	crocodileagent	47 779
7	tau	44 419
8	McCon	43 415

the 2010 tournament. *Mertacor-kNN* has only the k-NN capability, while it continues to bid with $\alpha = 0.3$. *Mertacor-RL* has the ability to adapt α , but is not equipped with k-NN. Last but not least, *Mertacor-Finals* combines all the modifications and extensions described in this paper and is the agent participated in the 2010 finals. In the first tournament, two agents from each one of the four versions of the agent competed, having all the same storage capacity equal to C^{MED} . In the second tournament the storage capacities were selected in competition terms (2 agents with C^{LOW} , 4 with C^{MED} and 2 with C^{HIGH}).

Results in Table 3 indicate that the extension added to the Mertacor-Finals version of the agent are giving the agent a small boost. Both versions were able to get the first two positions in this tournament. Even though the differences are not large enough and not statistical significant under paired t-testing, in tightly played competitions they could make the difference over ranking positions, like in 2010. Moreover, we believe that by optimizing these techniques, now implemented rather crudely, more profit is possible. It is also evident that including both extensions has more benefits than anyone of them alone.

Table 3: Average scores over 88 games of different versions of agent Mertacor with equal distribution capacity constraint $C = C^{MED}$.

Agent	Mean Score	Aggregate Score
Mertacor-Quals-1	\$53 330	\$53 036
Mertacor-Quals-2	\$52 742	
Mertacor-kNN-1	\$53 575	\$53 151
Mertacor-kNN-2	\$52 727	
Mertacor-RL-1	\$53 006	\$52 959
Mertacor-RL-2	\$52 912	
Mertacor-Finals-1	\$53 819	\$53 806
Mertacor-Finals-2	\$53 793	

When dealing with different capacities between games, the domain becomes more challenging, especially for on-line learning methods. In Table 4, one can find the results of the second tournament. Again the differences are not large enough, but there are small deviations between versions. The k-NN version is able to estimate better the capacity to be used by the agent and this benefits the final scoring of the agent. In Figure 3, one can observe the quality of the k-NN predictions. It is possible that by optimizing this algorithm, extremely ac-

curate results could be possible. As general comment, all versions of the agent, in both tournaments, were able to maintain their bank accounts to the level agent Mertacor scored in the finals. This is evidence of a robust strategy against both similar and different opponents.

Table 4: Average scores over 88 games of different versions of agent Mertacor in competition mode with respect to distribution capacity constraints.

Agent	Mean Score	Aggregate Score
Mertacor-Quals-1	\$53 679	\$53 289
Mertacor-Quals-2	\$52 900	
Mertacor-kNN-1	\$54 632	\$53 998
Mertacor-kNN-2	\$53 365	
Mertacor-RL-1	\$52 712	\$53 271
Mertacor-RL-2	\$53 831	
Mertacor-Finals-1	\$53 615	\$53 334
Mertacor-Finals-2	\$53 054	

6 Conclusions and Future work

In this report we have described our agent, *Mertacor*, for the TAC AA 2010 tournament. We have seen how empirical game theory can provide robust and effective results in a restricted real-world setting, when operationalized appropriately using simple abstractions. We were also able to discuss about the importance of the distribution constraint effect and the reserve score in our results, which significantly influenced our agent’s performance before and during the tournament. Last but not least, we have elaborated on two extensions, k-nearest neighbors and reinforcement learning that differentiate our agent from related work and provide added value to the agent with respect to making better predictions and adapting to the environment and the competition.

As future work, we would first need to use a more accurate user state predictor, such as the one implemented by TacTex. Moreover, we would like to extend our strategy to quadratic functions of the VPC, to incorporate the decrease in α for smaller corresponding valuations [Vorobeychik, 2009], as was also implemented in QuakTAC during the finals of 2009. Other non-linear function approximators could also be tested, under off-line parameter learning schemes for boosting time to converge to an optimal policy. Finally, it would be desirable to identify key parameters that influence more or less the optimal bidding percentage.

References

- [Berg *et al.*, 2010] Jordan Berg, Amy Greenwald, Victor Naroditskiy, and Eric Sodomka. A knapsack-based approach to bidding in ad auctions. In *Proceeding of the 2010 conference on ECAI 2010: 19th European Conference on Artificial Intelligence*, pages 1013–1014, Amsterdam, The Netherlands, The Netherlands, 2010. IOS Press.
- [Chang *et al.*, 2010] Meng Chang, Minghua He, and Xudong Luo. Designing a successful adaptive agent for tac ad auction. In *Proceeding of the 2010 conference on ECAI 2010*:

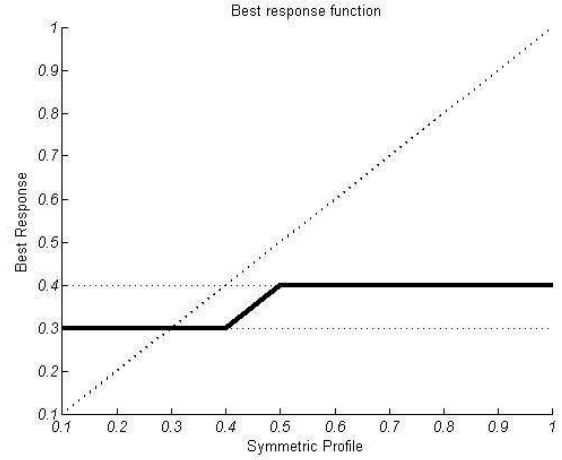


Figure 2: Best response function.

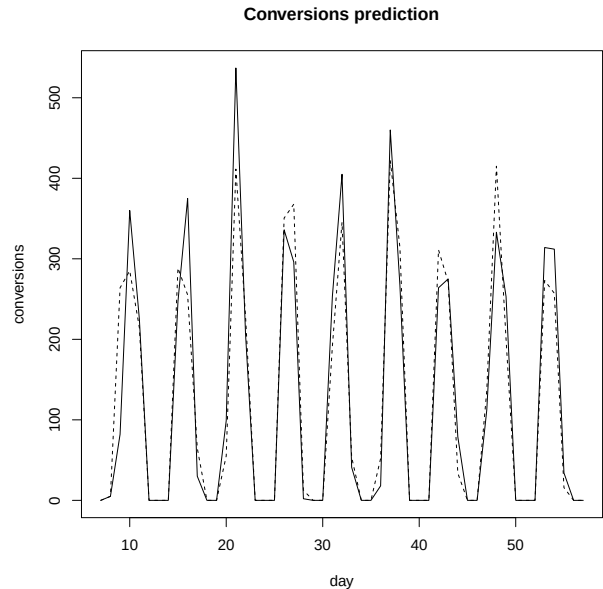


Figure 3: k-NN enabled predictions of conversions for the future day (dashed line). The solid line indicates the actual conversions made.

- 19th European Conference on Artificial Intelligence, pages 587–592, Amsterdam, The Netherlands, The Netherlands, 2010. IOS Press.
- [Cigler and Mirza, 2010] Ludek Cigler and Elias Mirza. EPFLAgent: A Bidding Strategy for TAC/AA. Presentation in Workshop on Trading Agent Design and Analysis (TADA), ACM EC 2010, July 2010.
- [Das et al., 2008] Aparna Das, Ioannis Goitis, Anna R. Karlin, and Claire Mathieu. On the effects of competing advertisements in keyword auctions. Working Paper, 2008.
- [Feldman and Muthukrishnan, 2008] Jon Feldman and S. Muthukrishnan. Algorithmic methods for sponsored search advertising. In Zhen Liu and Cathy H. Xia, editors, *Performance Modeling and Engineering*, 2008.
- [Jansen and Mullen, 2008] Bernard J. Jansen and Tracy Mullen. Sponsored search: an overview of the concept, history, and technology. *International Journal of Electronic Business*, 6(2):114–131, 2008.
- [Jordan et al., 2010] Patrick R. Jordan, Ben Cassell, Lee F. Callender, Akshat Kaul, and Michael P. Wellman. The ad auctions game for the 2010 trading agent competition. Technical report, University of Michigan, Ann Arbor, MI 48109-2121 USA, 2010.
- [Jordan, 2010] Patrick R. Jordan. *Practical Strategic Reasoning with Applications in Market Games*. PhD thesis, University of Michigan, Ann Arbor, MI, 2010.
- [Lahaie, 2006] Sébastien Lahaie. An analysis of alternative slot auction designs for sponsored search. In *EC '06: Proceedings of the 7th ACM conference on Electronic commerce*, pages 218–227, New York, NY, USA, 2006. ACM.
- [Mansour and Schain, 2010] Yishay Mansour and Mariano Schain. TAU Agent. Presentation in Workshop on Trading Agent Design and Analysis (TADA), ACM EC 2010, July 2010.
- [Munsey et al., 2010] M. Munsey, J. Veilleux, S. Bikkani, A. Teredesai, and M. De Cock. Born to trade: A genetically evolved keyword bidder for sponsored search. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pages 1–8, 2010.
- [Pardoe et al., 2010] David Pardoe, Doran Chakraborty, and Peter Stone. TacTex09: A champion bidding agent for ad auctions. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, May 2010.
- [PwC, 2010] PwC. IAB U.S. Internet Advertising Revenue Report for 2009, April 2010.
- [Sutton and Barto, 1998] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.
- [Vorobeychik, 2009] Yevgeniy Vorobeychik. Simulation-based game theoretic analysis of keyword auctions with low-dimensional bidding strategies. In *Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, 2009.
- [Vorobeychik, 2011] Yevgeniy Vorobeychik. A Game Theoretic Bidding Agent for the Ad Auction Game. In *Third International Conference on Agents and Artificial Intelligence (ICAART 2011)*, January 2011.

A Bidding Agent for Advertisement Auctions: An Overview of the CrocodileAgent 2010

Irena Siranovic, Tomislav Cavka, Ana Petric and Vedran Podobnik

University of Zagreb, Faculty of Electrical Engineering and Computing
Zagreb, Croatia

irenasiranovic@gmail.com, {tomislav.cavka, ana.petric, vedran.podobnik}@fer.hr

Abstract

Sponsored search is a popular form of targeted online advertising and the most profitable online advertising revenue format. Online publishers use different formats of unit price auctions to sell advertising slots. In the Trading Agent Competition Ad Auctions (TAC/AA) Game, intelligent software agents represent publishers which conduct keyword auctions and advertisers which participate in those auctions. The publisher is designed by game creators while advertisers are designed by game entrants. Advertisers bid for the placement of their ads on publisher's web page and the main challenge placed before them is how to determine the right amount they should bid for a certain keyword. In this paper, we present the CrocodileAgent, our entry in the 2010 TAC AA Competition. The agent's architecture is presented and a series of controlled experiments are discussed.

1 Introduction

Internet advertising provides significant income streams for online publishers. Revenue of major search engines such as Google¹, Yahoo!² and MSN³ amounts to tens of billions of dollars annually (e.g., in 2010, Google reported total advertising revenues over USD \$28 billion⁴). According to a report published by the Interactive Advertising Bureau⁵ and PricewaterhouseCoopers LLP⁶, sponsored search is the most profitable online advertising revenue format which accounted for 47% of the total Internet advertisement revenue in the USA in the first half of 2010⁷. In sponsored search (i.e., keyword advertising) publishers (i.e., search engines) use different formats [Feng *et al.*, 2007] of unit price auctions (e.g., keyword auctions) to sell advertising slots (i.e., positions in the list that contains search results) [Chen *et al.*, 2010].

¹<http://www.google.com/>

²<http://www.yahoo.com/>

³<http://www.msn.com/>

⁴<http://investor.google.com/financial/tables.html>

⁵<http://www.iab.net/>

⁶<http://www.pwc.com/>

⁷http://www.iab.net/media/file/IAB_report_1H_2010_Final.pdf

In a keyword auction advertisers bid for the placement of their ads (i.e., rank of the ad in the results of the sponsored search) which are then displayed on publisher's web page. The format of sponsored search results is very much alike the format of generic search results. Usually, it is comprised of the title of the ad, short description and a hyperlink to the advertiser's web page or the web page of the advertised product. An ad is chosen for a specific keyword(s) from the user's query, thus targeting users interested in advertiser's products.

A publisher conducts keyword auctions and solicits bids. When a user submits a query containing one or more keywords, sponsored ads are displayed to the user alongside the results of a generic search mechanism. At the end of an auction, the publisher ranks advertisers' bids (i.e., determines the placement of ads and cost-per-click (CPC) of those ads) [Easley and Kleinberg, 2010]. CPC is the price that an advertiser pays to the publisher each time its ad is clicked on.

According to earlier studies on user behaviour (e.g., [Joachims *et al.*, 2005]) the higher the position of the search result (e.g., ad, document) is, the more likely the users will click on it. This phenomenon, where the probability that the result will be clicked depends not only on its relevance, but also on the position of the result, is known as the position bias. Several models of the position bias have been proposed and the cascade model gave the best explanation for position bias in early ranks [Craswell *et al.*, 2008]. However, latest research has shown that 46% of users do not click sequentially (i.e., start from the best ranked result and continue to lower ranked ones) and 57% of them do not behave as suggested by the cascade model (i.e., first click on higher and afterwards on lower positioned results) [Jeziorski and Segal, 2010].

A challenge placed upon publishers evolves around the selection of a mechanism which will result with highest profits. There are two frequently used mechanisms for ranking solicited bids: i) rank-by-bid, and, ii) rank-by-revenue. As the name of the rank-by-bid mechanism states the bids are sorted in a descending order (i.e., bids offering higher CPC get higher ranked positions), while the rank-by-revenue mechanism multiplies the offered CPC with the ad's expected relevance (i.e., the percentage of users that will click on the ad once it is displayed to them) and afterwards sorts offers in a descending order of the calculated product [Lahaie, 2006]. In addition, some mechanisms offer advertisers the possibility to target a certain group of users by specifying the context

for viewing ads (e.g., user's location, time of day, etc.) and to control exposure by limiting the number of times an ad should be displayed to users [Lahaie *et al.*, 2008].

From the advertisers' point of view, the question is how to determine the right amount they should bid for a certain keyword(s) since the probability that their ad will be better ranked than other ads rises as they place higher bids [Varian, 2009]. The complexity of the answer to this question increases as the availability of information about user behaviour, as well as bidding behaviour of other advertisers decreases.

The paper is organized as follows. Section 2 describes the Trading Agent Competition / Ad Auctions (TAC/AA) game characteristics. A brief overview of research in the area of TAC/AA bidding strategies is given in Section 3. Section 4 presents the CrocodileAgent 2010, our entrant for the 2010 TAC/AA Tournament. Section 5 presents the conducted experiments and the obtained results, while Section 6 concludes the paper and gives an outline for future work.

2 Trading Agent Competition / Ad Auctions

Researchers test advertisers' bidding strategies by using the designed market simulators which provide a risk free environment [Acharya *et al.*, 2007]. The TAC/AA game [Jordan *et al.*, 2009; Jordan and Wellman, 2010], which was released in 2009, is based on such a market simulator. The game enables advertisers to bid on multiple queries and to define budget constraints in an information-lacking and competitive environment. Eight intelligent program agents which represent online advertisers participate in the game. Each advertiser sells 9 different products which are specified by a manufacturer (Flat, Lioneer and PG) and a component (TV, Audio and DVD). Furthermore, each advertiser is specialized for one manufacturer and for one component type.

The ad ranking mechanism varies between rank-by-bid and rank-by-revenue [Lahaie and Pennock, 2007] and it is chosen at the beginning of each game. Users can generate 16 different queries: specifying both manufacturer and component (i.e., F2 level query), specifying only manufacturer or component (i.e., F1 level query), or neither (i.e., F0 level query). Correspondingly, conversion probability increases with more keywords specified. Users can find themselves in one of the following states: i) non-searching, ii) searching, and, iii) transacted; while user population in each state (and sub-state) is modelled as a Markov chain.

In the TAC/AA game scenario, when a user submits a query, an auction for the given keyword(s) starts. The auction is an instance of the repeated generalized second price auction (i.e., the price that the advertiser pays for the position of its ad is determined by the price which did the winner of the next-best position offered in its bid) and the first position is allocated to the bidder with the highest bid. An advertiser sends a bid bundle (i.e., one bid for each query class) to the publisher every day. A bid consists of: i) the CPC an advertiser is ready to pay, ii) the chosen ad which can be generic or targeted (i.e., specified manufacturer and/or component), iii) budget limits for each query class, and, iv) budget limit for all queries altogether for the following day

(optional). The publisher uses the information from the bid when it runs ad auctions for received user queries. Advertisers receive daily reports about the outcomes of the prior (i.e., two days old) auctions and use those information to generate new bids. Daily reports include: i) query report, ii) account status report, and, iii) sales report. The game lasts 60 virtual days.

3 Related Work

The bidding strategy for keyword auctions has been a great challenge for researchers that conducted various simulations and empirical analyses on this matter. [Berg *et al.*, 2010] have presented autonomous bidding strategies for ad auctions based on click probability and CPC estimations. The algorithms for bid optimization come in two parts: i) rule-based algorithms, and, ii) greedy multiple choice Knapsack algorithms. [Pardoe and Stone, 2010] have shown a particle filter that can be used for estimating other agents' bids given a periodic ranking of their bids. The particle filter uses bidding behaviour models of other advertisers. Since the information revealed about competitor advertisers is limited, they have shown how such models can learn from past bidding data. [Cigler, 2009] has described several bidding strategies (Return-on-investment (ROI), Knapsack ROI, Balanced Best-response, online Knapsack) and evaluated their performance empirically and against omniscient strategy. The best performing strategies were ROI and Balanced Best-response. Furthermore, Cigler presented a new profit maximizing strategy for multiple keyword ad campaigns. The strategy takes into account the budget constraint which has shown to be a competitive strategy particularly for small budgets.

4 CrocodileAgent 2010

With an intention to better investigate strategic approaches to ad auctions mechanisms, an advertiser agent CrocodileAgent has been designed. A bidding strategy was chosen among several strategies (profit maximization, linear regression [Witten and Frank, 2005] and analysed through controlled experiments. The strategy was compared with already existing in literature and practice by running TAC/AA games. As a result, the CrocodileAgent setup which produced most satisfactory performance was identified. The Crocodile Agent's bidding strategy for participating in ad auctions can be divided into three logical segments: i) the ad generator, ii) the CPC generator, and, iii) the daily spend limit generator. Figure 1 shows the bidding model.

4.1 The ad generator model

The algorithm for ad generation is shown in Algorithm 1. The chosen method implements ad generation based on query focus levels. This method was chosen by analysing final results in game simulations with several different implemented algorithms for ad generation. The observed conversion probability is the highest when user submits a query with focus level F2. Since there are 9 different products and each user has a preference for just one, it is highly probable that a user will click on the ad that matches the query when he/she submits an F2 query. Furthermore, we have concluded that if an agent

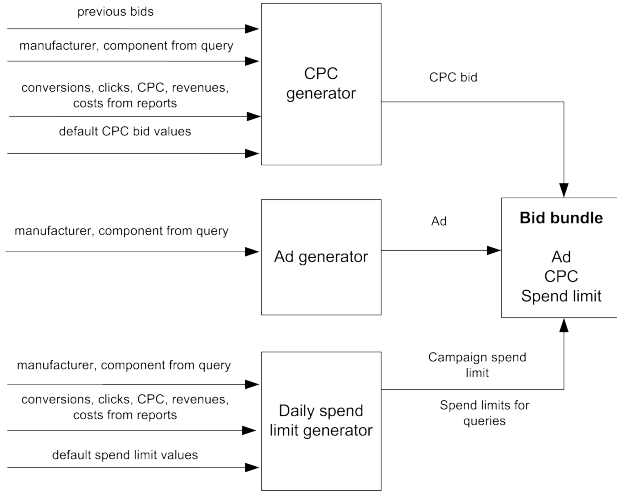


Figure 1: CrocodileAgent's bidding model

provides a generic ad for submitted F2 query, the probability that user clicks on the ad decreases significantly, especially if competitor agents choose a targeted advertisement.

Algorithm 1: CrocodileAgent's ad generation algorithm

```

if query focus level = F0 or F1 then
    generated ad = generic
else
    generated ad =
        targeted (F2 manufacturer, F2 component)
end

```

For a submitted query F1, CrocodileAgent generates generic ads - if CrocodileAgent generated targeted ad, it would have to decide which component or manufacturer to add alongside specified keyword (i.e., manufacturer or component, respectively). The negative aspect of the latter approach is that targeted ad has to match user preference in order to agent receive a conversion (a probability that generated ad matches user preference is 1/3). If the ad does not match, click probability decreases according to the targeting factor [Jordan *et al.*, 2009], and vice versa. Moreover, if the ad does not match user preferences, the user may click on the ad, but will not convert and thus increase only the CPC. On the other hand, the benefit of the approach that results with targeted ad manifests in a case when ad matches submitted query - positive aspects of this approach are: i) additional profit gained by component (CSB) or manufacturer specialization bonus (MSB), ii) increased odds of converting based on CSB, and, iii) increased targeting effect.

4.2 The CPC generator model

The CPC bid is defined for every query type with an intention to maximize the profit (profit = revenue - cost). Taking into consideration the two days delay of reports, the algorithm for generating the CPC for the first two days of the game contains fixed values of CPC. Values for all query types are defined

Algorithm 2: CrocodileAgent's initial bidding algorithm

```

result = match between query and agents specialties;
switch result do
    case miss
        bid =  $\alpha$ ;
    case miss-neutral
        bid =  $\beta$ ;
    case miss-hit
        bid =  $\gamma$ ;
    case neutral
        bid =  $\delta$ ;
    case neutral-hit
        bid =  $\epsilon$ ;
    case hit
        bid =  $\xi$ ;
endsw
where  $\beta < \gamma < \alpha \leq \delta \leq \epsilon < \xi$ ;

```

based on matches between component and manufacturer in a query and agent specialties as shown in Algorithm 2.

The more accurate the match is, the higher is the bid. Additionally, values are scaled by agent's capacity (i.e., the bid is slightly decreased in cases of medium and low capacity). Definitions of matching values are listed in Table 1, while Table 2 contains the parameter values. All parameter values were optimized using the heuristic approach. The decisions were based on conducted game simulations (locally and in the TAC/AA 2010 qualifying rounds) by analysing and comparing the CrocodileAgent's profit for different sets of parameters. The used decision making method could be best compared to the bisection method.

A method for defining CPC bids in the rest of the game is based on calculating the conversion rate (the ratio of average

Table 1: Definitions of specialty matching values

Value	Query type	Definition
miss	F2	Component and manufacturer from query do not match agent's specialties.
miss-neutral	F1	Component or manufacturer from query does not match agent's specialties, second keyword is null.
miss-hit	F2	Component or manufacturer from query does not match agent's specialties, second keyword is a match.
neutral	F0	A query without specified manufacturer and component.
neutral-hit	F1	Component or manufacturer from query matches agent's specialties, second keyword is null.
hit	F2	Component and manufacturer from query are agent's specialties.

Table 2: Optimal value of parameters for initial bidding

Parameter	α	β	γ	δ	ϵ	ξ
Value	1.15	0.65	1.05	1.15	1.15	1.25

number of clicks to conversions). Furthermore, the bid is defined in a dependency to the conversion rate. If the rate is satisfactory, the new bid is based on the CPC bid from two days ago. On the contrary if the rate is not satisfactory, the new bid is based on the CPC that an agent actually paid, received on the latest report. The bid is later adjusted depending

Algorithm 3: CrocodileAgent’s ad generation algorithm

```

revenue = last revenue for a query received in report;
nclick = average number of clicks per day;
nconversion = average number of conversions per day;
queryfl = query focus level;
result = match between query and agent’s specialties;
if revenue  $\neq$  0 then
    mod = nclick/nconversion;
    bid = DefineBid(mod);
    if queryfl == F0 or queryfl == F1 then
        bid = focus level parameter · bid
    end
    if result == hit then
        bid = specialty parameter · last paid bid
    end
else
    if queryfl == F2 then
        bid = min(minimum bid zero,
                    hit parameter · last bidday-2);
        if
            manufacturer and component of query ==
            agent’s specialties then
                bid = specialty parameter zero · bid
            end
        else
            bid = parameter zero · last bidday-2;
        end
    end
end

function DefineBid(mod)
if mod < ratio lower bound then
    bid = max(minimum bid,
               decrease factor · last bidday-2);
else if
    ratio lower bound ≤ mod ≤ ratio middle value then
        bid = max(minimum bid,
                   increase factor · last paid bid);
else
    bid = max(minimum bid,
               max increase factor · last paid bid);
end
return bid;

```

Table 3: Optimal value of bidding parameters during a game

Parameter	Definition	Value
ratio lower bound		10
ratio middle value		0.65
minimum bid	minimum for CPC that an agent will bid	0.10
decrease factor	decrease factor in case of very good <i>mod</i>	0.97
increase factor	increase factor in case of average <i>mod</i>	1.15
max increase factor	increase factor in case of very bad <i>mod</i>	1.20
focus level parameter		0.80
specialty parameter		1.30
minimum bid zero	minimum for CPC that an agent will bid in case that last revenue was zero	0.30
hit parameter		1.30
specialty parameter zero		1.20
parameter zero	a parameter used when agent’s last revenue was zero	1.05

on the query focus level and the CrocodileAgent’s specialties. The CPC generator model is shown in Algorithm 3. The last part of the CPC bidding model is adjusting the bid according to the agent’s capacity (i.e., if the capacity is low, the bid decreases).

In defining CPC bids the most important aspect is the “quality” of an ad, which is measurable through agent’s profit. Quality can be defined as an estimated value that a click event will turn into conversion, which is based on previous report data. Note that CrocodileAgent’s bidding strategy does not change during the game. Table 3 contains the parameters that gave the most satisfactory results in a con-ducted experiment.

4.3 The spend limit manager

General spend limit

Too many conversions lead to decrease of possible conversions in the future due to stock shortage, as defined by the game rules [Jordan and Wellman, 2010]. The stock managing policy is necessary in a scenario when a certain amount of product must be immediately available in order to avoid profit loss. On the other hand, excessive product storage leads to “dead capital”. Additionally, the other driver for using spend limits are the clicks of informational searchers (F0).

The general spend limit manager algorithm is shown in Algorithm 4, while corresponding optimal parameter values are listed in Table 4. In the first five days of the game the limit is fixed and determined based on agent’s capacity. In the controlled experiment we have concluded that the lack of spend limit has the most significant (negative) impact on Crocodile Agent’s profit when the capacity is low.

Algorithm 4: General spend limit manager

```

if first five days then
  switch capacity do
    case low
      spend limit = spend limit low fixed;
    case medium
      spend limit = spend limit medium fixed;
    case high
      spend limit = spend limit high fixed;
  endsw
else
  switch capacity do
    case low
      spend limit = spend limit low;
    case medium
      spend limit =
        min(spend limit medium, custom profit);

    case high
      spend limit =
        min(spend limit high, custom profit);
  endsw
end

```

Table 4: Optimal value of parameters for general spend limit

Parameter	Min
spend limit low fixed	700
spend limit medium fixed	1000
spend limit high fixed	1200
spend limit low	750
spend limit medium	1150
spend limit high	1350

Query spend limit

The query spend limit manager defines a mechanism for optimal distribution of investments in order to maximize the profit. The mechanism ensures that the click number is limited based on the calculated quality of an ad. Considering the report delay, the spend limits for the first two days are fixed. They are generated based on matching the CrocodileAgent’s specialties with keywords in query. In the rest of the game spend limit is defined according to the click and conversion ratio, as well as previous profits. Low ratio corresponds to

Table 5: Optimal value of parameters for general spend limit

Parameter	Value	Parameter	Value
a	60	ratio lower bound	5
b	80	ratio middle value	10
c	100	minimum limit	20
d	120	default limit	40
e	140	neutral	1.10
f	180	neutral-hit	1.20
		hit	1.30

Algorithm 5: Query spend limit manager

```

nclick = average number of clicks per day;
nconversion = average number of conversions per day;
mod = nclick/nconversion;
result = a match between query and agent’s specialties;
if first two days then
  switch result do
    case miss
      limit = a;
    case miss-neutral
      limit = b;
    case miss-hit
      limit = c;
    case neutral
      limit = d;
    case neutral-hit
      limit = e;
    case hit
      limit = f;
  endsw
else
  if mod < ratio lower bound then
    limit = min (minimum limit, revenue/2);
  else if
    ratio lower bound ≤ mod ≤ ratio middle value
  then
    limit = min (minimum limit, revenue/3);
  else
    limit = default limit;
  end
  limit = SpecialtyMatchingLimit (result,
    limit)
end

function SpecialtyMatchingLimit (result, limit)

  switch result do
    case neutral
      limit = neutral · limit;
    case neutral-hit
      limit = neutral – hit · limit;
    case hit
      hit : limit = hit · limit;
  endsw

```

the successful advertisement. Therefore, the lower is the ratio, the higher is the limit. The query spend limit manager algorithm is shown in Algorithm 5, while corresponding optimal parameter values are listed in Table 5.

5 Controlled experiment

In order to evaluate performance of the CrocodileAgent, which placed 6th in the TAC/AA 2010 Competition Finals, an experiment was conducted by repeating games with fixed parameters. Based on the analysis of the results CrocodileAgent’s deficiencies were identified and guidelines for future improvements were set. The participants in the

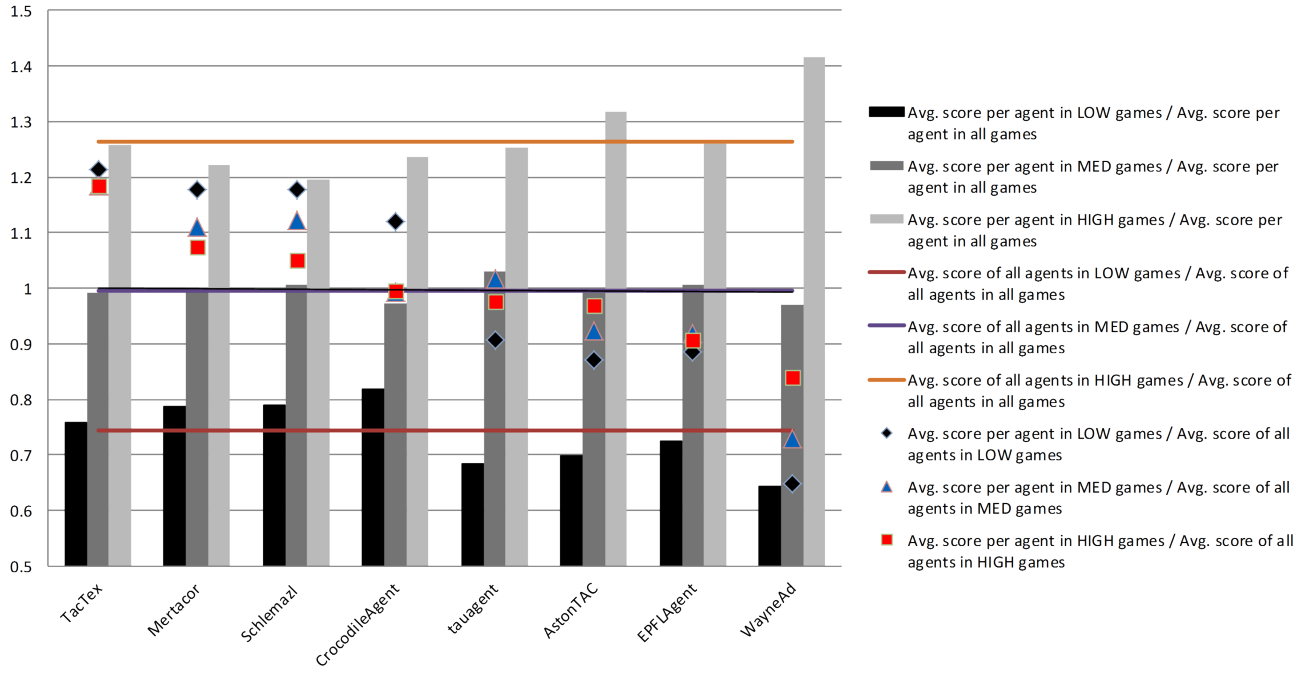


Figure 2: Agents' relative profit with respect to assigned capacity

experiment were the following agents which competed in the TAC/AA 2010 Competition *TacTex*, *Mertacor*, *Schlemazl*, *CrocodileAgent*, *tauagent*, *EPFLAgent*. Additionally, due to lack of TAC/AA 2010 agents in the official agent repository, two agents from TAC/AA 2009 Competition, *AstonTAC* and *WayneAd*, were included in experiment. The average results from 40 games played are shown in Table 6.

The games in the controlled experiment were configured to ensure fair capacity distribution among competing agents each agent played with high capacity ten times, twenty times with medium capacity and ten times with low capacity. In a specific game, two agents had low capacity, two had high capacity and four of them had medium capacity. The goal of the competition was to observe agents' behaviour in respect with the assigned capacities. The results of these observations are shown in Figure 2.

In the graph from Figure 2 bars represent the ratio between the average result of a single agent in games with the spec-

ified capacity (i.e., high, medium and low) and the average result of the same agent in all games. We call this ratio the intra-agent relative profitability. On the other hand, the horizontal lines represent the ratio between average results of all agents in games with the specified capacity (i.e., high, medium and low) and the average results of all agents in all games. This measure represents the average intra-agent relative profitability of all agents. Finally, squares, triangles and diamonds represent the ratio between the average result of a single agent in games with the specified capacity (i.e., high, medium and low, respectively) and the average result of all agents in games with the same capacity. This graph allows us to compare single agent's average profit achieved in games with different assigned capacities with: i) its average profit in all games, and, ii) the average profit of all agents' in games with the same capacity. While the former measure enables us to compare the profitability of agent's strategies in games with different capacities with the agent's overall profitability, the latter measure provides relative benchmarking among different agents.

If we take a look at the average values of profits achieved in games with low capacity, we can notice that the *CrocodileAgent* has the highest intra-agent relative profitability among all agents in the controlled experiment. However, the *CrocodileAgent* has smaller intra-agent relative profitability than the average intra-agent relative profitability of all agents both in medium and high capacity games. At the same time, we can also notice that, when comparing the average score of different agents, the *CrocodileAgent* has 12% better score than all agents' average in games with low capacity, while its performance in both medium and high capacity games is equal to all agents' average.

Table 6: Average results in the conducted competition

Position	Agent	Game score
1.	TacTex	57 848
2.	Mertacor	53 998
3.	Schlemazl	53 933
4.	CrocodileAgent	49 435
5.	tauagent	47 789
6.	AstonTAC	45 104
7.	EPFLAgent	44 179
8.	WayneAd	36 456

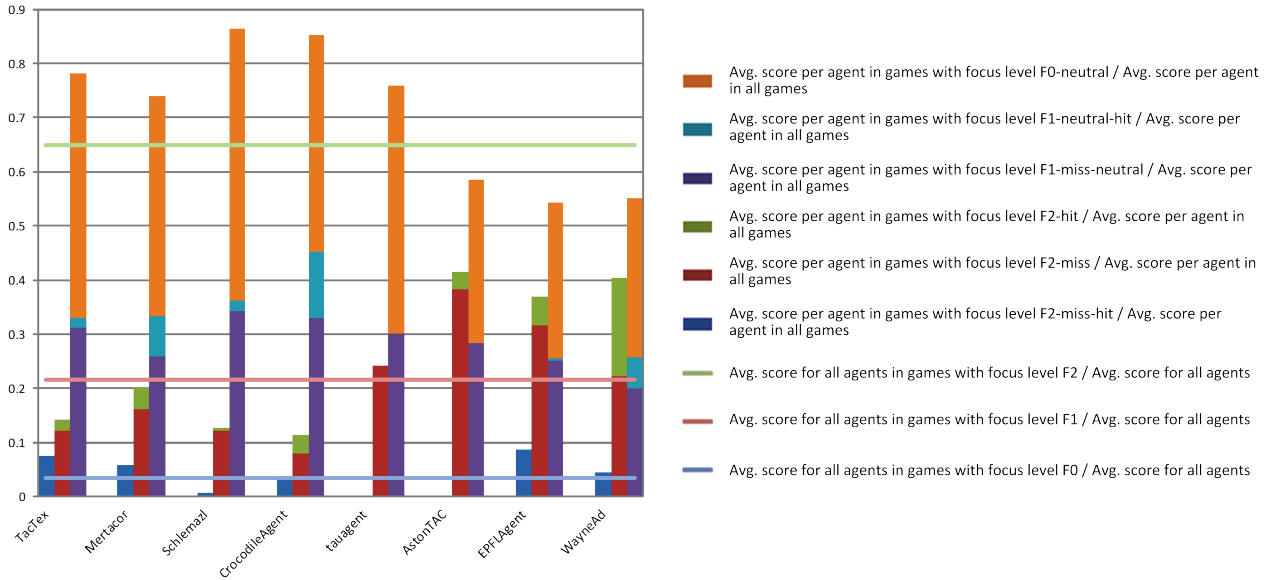


Figure 3: Profit distribution among query classes

From this analysis we can identify certain CrocodileAgent’s deficiencies improvement of those CrocodileAgent’s drawbacks in future versions could significantly increase its profits. Namely, we can conclude that the CrocodileAgent should examine the possibility of using other strategies for achieving profits in medium and high capacity games in order to increase its profit in those games. Possibly, such an approach could significantly (positively) affect the final result of the CrocodileAgent in the TAC/AA 2011 competition.

Another interesting fact we can learn from Figure 2 is that the relative intra-agent profitability of the TacTex agent, the best agent in competition, is approximately equal to the average intra-agent profitability of all competing agents (for all three capacity allocations). Furthermore, it is attention-grabbing to note that the WayneAd agent, who placed last in the competition, has the highest relative intra-agent profitability in games with high capacity. However, WayneAd’s weakest absolute results in medium and low capacity games are the reason for its the last place in the competition.

After analysing the impact of capacity on agents’ achieved profit, we have also analysed the correlation of the achieved profit and the query category. As mentioned earlier, there are three types of queries (i.e., F0, F1 and F2) that users generate. Each advertiser selects an ad for display for each query type, choosing between generic or targeted ad mentioning a particular product [Jordan *et al.*, 2009]. The agents’ profits from all games in the experiment were grouped based on the type of user query. The graph presented in Figure 3 shows the distribution of profit achieved from transactions originating from different query types.

In the graph from Figure 3, the horizontal lines represent the ratio between average results of all agents for queries of specified type (i.e., F0, F1 and F2) and the average results of all agents for all types of queries. On the other hand, the bars

represent the ratio between agent’s average result for queries of specified type (i.e., F0, F1 and F2) and agent’s average result for queries of all types.

If we look at the average values of profits originating from different query types, we can notice that agents, who have better final result, also achieve higher profits on targeted ads. The last three agents in the competition AstonTAC, EPFLAgent and WayneAd achieved the lowest relative profits from focus level F2 queries and the highest relative profits from focus level F1 queries.

Expressed in percentages and in co-relation to average results for queries of all types, TacTex agent made 78% of its total profit from focus level F2 queries. Consequently, TacTex achieved 22% of its total profit from focus level F0 and focus level F1 queries. Agent Schlemazl, who has the highest relative score for F2 queries, achieved 86% of its total profit from focus level F2 queries.

Another interesting fact we can learn from Figure 3 is that overall profits are not highly correlated with the fraction of clicks received under manufacturer specialty (i.e., focus level F2-hit). Therefore, we conclude that shifting spend towards queries focusing on manufacturer specialty is not a guarantee of greater profitability for agents, despite bonuses they get if such transactions take place.

If we analyse profit distribution for CrocodileAgent, depending on the type of user query, we can notice that CrocodileAgent is achieving approximately 85% of its total profit from focus level F2 queries. This percentage is higher for CrocodileAgent than for two best-placed agents, TacTex and Mertacor. On the other hand, CrocodileAgent achieved lower relative profits than TacTex and Mertacor from focus level F1 queries. We can conclude that CrocodileAgent should try to redistribute a few percent of its profit obtained from F2-queries to F1-queries, while maintaining the share of F0-queries.

6 Conclusion and future work

Trading Agent Competition Ad Auctions (TAC/AA) Game enables academic community and advertising industry to analyse effects of various bidding strategies through simulation of sponsored search scenarios. The fact that sponsored search is the most profitable online advertising revenue format gives great importance to ad auction research as well.

In this paper, we presented bidding strategies of the CrocodileAgent, the representative of University of Zagreb in the TAC/AA 2010 Competition. Furthermore, we conducted controlled experiment with best-ranked agents from 2010 and 2009 TAC/AA Finals. Based on an analysis of the controlled experiment, we: i) explained some reasons why certain agents performed better than others, and, ii) identified mechanisms that should be improved in order to boost CrocodileAgent's performance.

For future work we plan to enhance CrocodileAgent's performance by implementing guidelines for improvements defined based on controlled experiment analysis. Namely, we will: i) redesign strategies for achieving profits in medium and high capacity games, and ii) redistribute a part of relative CrocodileAgent's profit from F2-queries based profit to F1-queries based profit.

Acknowledgments

The authors acknowledge the support of research project "Content Delivery and Mobility of Users and Services in New Generation Networks" (036-0362027-1639), funded by the Ministry of Science, Education and Sports of the Republic of Croatia.

References

- [Acharya *et al.*, 2007] Soam Acharya, Prabhakar Krishnamurthy, Ketan Deshpande, Tak Yan, and Chi-Chao Chang. A Simulation Framework for Evaluating Designs for Sponsored Search Markets. In *16th International World Wide Web Conference*, May 2007.
- [Berg *et al.*, 2010] Jordan Berg, Amy Greenwald, Victor Naroditskiy, and Eric Sodomka. A first approach to autonomous bidding in ad auctions. In *EC 2010 Workshop on Trading Agent Design and Analysis, TADA 2010*, New York, NY, USA, 2010. ACM.
- [Chen *et al.*, 2010] Jianqing Chen, Juan Feng, and Andrew B. Whinston. Keyword auctions, unit-price contracts, and the role of commitment. *Production and Operations Management*, 19(3):305–321, 2010.
- [Cigler, 2009] Ludek Cigler. *Semester Project: Bidding Agent for Advertisement Auctions*. Ecole Polytechnique Federale de Lausanne, 2009.
- [Craswell *et al.*, 2008] Nick Craswell, Onno Zoeter, Michael Taylor, and Bill Ramsey. An experimental comparison of click position-bias models. In *Proceedings of the international conference on Web search and web data mining, WSDM '08*, pages 87–94, New York, NY, USA, 2008. ACM.
- [Easley and Kleinberg, 2010] D. Easley and J. Kleinberg. *Networks, Crowds, and Markets: Reasoning About a Highly Connected World*. Cambridge University Press, 2010.
- [Feng *et al.*, 2007] Juan Feng, Hemant K. Bhargava, and David M. Pennock. Implementing sponsored search in web search engines: Computational evaluation of alternative mechanisms. *INFORMS Journal on Computing*, 19(1):137–148, 2007.
- [Jeziorski and Segal, 2010] Przemyslaw Jeziorski and Ilya Segal. What makes them click: Empirical analysis of consumer demand for search advertising. Economics Working Paper Archive 569, The Johns Hopkins University, Department of Economics, August 2010.
- [Joachims *et al.*, 2005] Thorsten Joachims, Laura Granka, Bing Pan, Helene Hembrooke, and Geri Gay. Accurately interpreting clickthrough data as implicit feedback. In *Proceedings of the 28th ACM SIGIR conference on Research and development in information retrieval, SIGIR '05*, pages 154–161, New York, NY, USA, 2005. ACM.
- [Jordan and Wellman, 2010] Patrick R. Jordan and Michael P. Wellman. Designing an ad auctions game for the trading agent competition. In *Agent-Mediated Electronic Commerce. Designing Trading Strategies and Mechanisms for Electronic Markets*, volume 59 of *Lecture Notes in Business Information Processing*, pages 147–162. Springer Berlin Heidelberg, 2010.
- [Jordan *et al.*, 2009] Patrick R. Jordan, Ben Cassell, Lee F. Callender, and Michael P. Wellman. The ad auctions game for the 2009 trading agent competition. Technical report, 2009.
- [Lahaie and Pennock, 2007] Sébastien Lahaie and David M. Pennock. Revenue analysis of a family of ranking rules for keyword auctions. In *Proceedings of the 8th ACM conference on Electronic commerce, EC '07*, pages 50–56, New York, NY, USA, 2007. ACM.
- [Lahaie *et al.*, 2008] Sébastien Lahaie, David C. Parkes, and David M. Pennock. An expressive auction design for online display advertising. In *Proceedings of the 23rd national conference on Artificial intelligence - Volume 1*, pages 108–113. AAAI Press, 2008.
- [Lahaie, 2006] Sébastien Lahaie. An analysis of alternative slot auction designs for sponsored search. In *Proceedings of the 7th ACM conference on Electronic commerce, EC '06*, pages 218–227, New York, NY, USA, 2006. ACM.
- [Pardoe and Stone, 2010] David Pardoe and Peter Stone. A particle filter for bid estimation in ad auctions with periodic ranking observations. In *EC 2010 Workshop on Trading Agent Design and Analysis, TADA 2010*, Cambridge, Massachusetts, 2010. ACM.
- [Varian, 2009] Hal R. Varian. Online ad auctions. *American Economic Review*, 99(2):430–34, 2009.
- [Witten and Frank, 2005] Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Publishers, San Francisco, CA, 2005.