

Stronger CDA Strategies through Empirical Game-Theoretic Analysis and Reinforcement Learning

L. Julian Schwartzman and Michael P. Wellman

University of Michigan, Artificial Intelligence Laboratory
Ann Arbor, MI 48109-2121 USA
{lschvart,wellman}@umich.edu

ABSTRACT

We present a general methodology to automate the search for equilibrium strategies in games derived from computational experimentation. Our approach interleaves empirical game-theoretic analysis with reinforcement learning. We apply this methodology to the classic Continuous Double Auction game, conducting the most comprehensive CDA strategic study published to date. Empirical game analysis confirms prior findings about the relative performance of known strategies. Reinforcement learning derives new bidding strategies from the empirical equilibrium environment. Iterative application of this approach yields strategies stronger than any other published CDA bidding policy, culminating in a new Nash equilibrium supported exclusively by our learned strategies.

Categories and Subject Descriptors

I.2.6 [Artificial Intelligence]: Learning; I.2.11 [Artificial Intelligence]: Multiagent systems; J.4 [Social and Behavioral Sciences]: Economics

General Terms

Economics, Experimentation

Keywords

Continuous double auction, empirical game-theoretic analysis, trading agents

1. INTRODUCTION

The continuous double auction (CDA) [6] is a dynamic market environment, where bidders repeatedly exchange offers to buy and sell units of a good, in order to maximize trade surplus. Bidding in CDAs is a challenging strategic problem, addressed by much prior agent-based research. The popularity of CDAs as a research domain can be attributed in large part to real-world salience: trillions of dollars' worth of financial securities and commodities contracts are traded through versions of CDAs annually. Researchers are also drawn to the challenge of strategic reasoning in a dynamic and uncertain environment. The complexity of this domain (huge strategy space, many agents, severely incomplete and incrementally revealed information) has generally precluded analytic solu-

Cite as: Stronger CDA Strategies through Empirical Game-Theoretic Analysis and Reinforcement Learning, L. Julian Schwartzman and Michael P. Wellman, *Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009)*, Decker, Sichman, Sierra and Castelfranchi (eds.), May, 10–15, 2009, Budapest, Hungary, pp. XXX-XXX. Copyright © 2009, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

tion. Hence, agent-based simulation experiments have provided the primary source of evidence for comparing strategy ideas to date.

A particular difficulty in experimental analysis of strategic domains is that the efficacy of an agent's strategy can be highly dependent on strategy choices of other agents. Previous CDA studies have dealt with this issue in a variety of ways, including fixed distributions of strategies (e.g., uniform), factorial designs, and round-robin tournaments. In the absence of dominant strategies, such treatments offer useful evidence but strain to support definitive conclusions. Comparisons based on evolutionary stability analysis [1, 4, 17, 30] provide additional confidence due to the pressures exerted on the agent population by the selection dynamic.

In the *empirical game-theoretic analysis* (EGTA) approach, simulation experiments provide a statistical basis for estimating an approximate game, and standard game-theoretic concepts are employed to evaluate strategy profiles. EGTA methods accumulate knowledge about strategy performance across the range of other-agent contexts through directed Monte Carlo sampling. By systematically adding strategy candidates to the mix, we can produce a comprehensive model of a broad strategy space, amenable to game-theoretic analysis. Previous applications of EGTA to market environments yielded insights about strategies developed for trading agent competitions and simultaneous auctions [11, 35].

In the work reported here, we undertook a systematic EGTA analysis of the explored CDA strategy space, and extended the methodology to automatically generate new strategy candidates using reinforcement learning (RL). Our analysis of prior strategies constitutes the most comprehensive CDA experiment to date, and confirms the superiority of GD and extended GD strategies (see Section 3.4) compared to previous proposals. By interleaving EGTA with RL, we further show how to improve over previously identified strategies, by deriving new bidding policies that outperform all of these in equilibrium. Iterating this approach, we converge on a new mixed strategy that may now be considered the "reigning champion" for this instance of the CDA domain. Although these RL-derived strategies are somewhat opaque, regression analysis sheds some light on the features that distinguish their behavior.

As important as the new strategies themselves is the general method for deriving them for a particular CDA environment. The interleaved EGTA/RL approach provides a framework for generating improved strategies for other CDA configurations, as well as a wide range of other challenging strategic domains.

2. CONTINUOUS DOUBLE AUCTION

The CDA is one of the most ubiquitous auction types, underpinning most financial and commodity exchanges. *Double* means that both buyers and sellers bid, and *continuous* means that bid matches are cleared as soon as they occur. Upon receiving a new sell (buy)

bid, the auction checks whether it matches the best outstanding buy (sell) bid. If so, the respective bidders trade at a price usually determined by the earlier bid, and a new *price quote* is issued. The typical price quote consists of a *BID-ASK* pair, where *BID* is the highest outstanding purchase price (the price at which one could sell at least one unit), and *ASK* is the lowest outstanding sell price.

Standard CDAs allow bidding for multiple units of a good, and these units are commonly treated as divisible quantities. As a result, bids effectively express willingness to trade any fraction of the specified quantity, at the given price. Unmatched or partially matched bids are listed in the *order book*, and typically remain outstanding until they are matched, replaced, withdrawn, or canceled when the auction closes. Variations of this general scheme are plentiful, with different bidding, clearing, and quoting rules.

3. EXISTING STRATEGIES

Double auctions have been the subject of extensive experimental investigation. The first effort to compare a range of automated bidding strategies was the 1990 Santa Fe Double Auction Tournament (SFDAT) [6]. Here we provide brief descriptions of the major CDA strategies proposed in the literature to date, with references to the original publications for further details.

3.1 Kaplan

The winner of SFDAT was a *sniping* strategy submitted by Todd Kaplan [22]. The basic idea of Kaplan is to keep quiet for most of the auction, then attempt to “steal the deal” in the last opportunity if it is expected to produce some gain. By remaining inactive while other bidders conduct price discovery, the agent avoids revealing any private information. Despite its surprising success at SFDAT, subsequent studies found strong incentives to deviate to other strategies from an all-Kaplan population [33, 28]. Our implementation of Kaplan follows that of Tesouro and Das [28].

3.2 Zero Intelligence

Zero intelligence (ZI) agents were proposed by Gode and Sunder [9], primarily as an attempt to demonstrate that convergence to equilibrium prices was due to the structure of the CDA and independent of the traders’ motivation, intelligence, or learning. ZI agents simply generate bids at random prices drawn from a uniform distribution. Despite the simplicity of the strategy, Gode and Sunder showed that a market consisting of ZI traders achieved high allocation efficiency and quickly converged to equilibrium prices.

We employ a flavor of the strategy called ZI-c (ZI with constraint), restricting the range of prices to those that would not generate a loss. We also consider a variant that we call ZIbtq, which further constrains prices to those that would beat the current price quote.

3.3 Zero Intelligence Plus

Observing that markets slightly more complex than the model of Gode and Sunder resulted in inefficient outcomes for ZI agents, Cliff [3] proposed zero intelligence “plus” (ZIP) agents to remedy the problem. The ZIP strategy uses an elementary form of machine learning, adjusting profit margins based on market conditions. Sellers raise their expected margins when trade prices are above the seller’s current offer, and decrease them if trade or *ASK* prices are below it. Buyers employ an analogous criterion.

Since the original paper, ZIP has been widely tested and in some cases improved. One such improvement was proposed by Tesouro and Das [28], who used an array of profit margins (one per traded unit) instead of a single value. Another modification was suggested by Preist [18], who used a simpler update rule. Finally, Cliff [4]

also described “ZIP60”, a parameterized version of the original ZIP agent that included 60 parameters tuned with a genetic algorithm. Our implementation of ZIP follows that of Tesouro and Das.

3.4 Gjerstad and Dickhaut

A particularly successful bidding strategy, now generally called *GD*, was proposed by Gjerstad and Dickhaut [8]. Based on historical data from the auction, GD maintains a *belief function* representing the probability of a bid being accepted depending on its price. For a seller, the belief function is given by

$$\Pr(p) = \frac{TS(p) + B(p)}{TS(p) + B(p) + US(p)}, \quad (1)$$

where $TS(p)$ is the total number of transacted sell bids at a price p or higher, $B(p)$ is the total number of buy bids submitted to the auction at a price p or higher, and $US(p)$ is the total number of unmatched sell bids up to p . These values are calculated by taking into account the history of bids (submitted by all agents) leading to the last M transactions, and extended to the positive reals using cubic spline interpolation. The agent bids at a price p that maximizes expected surplus, defined as $\arg \max_p \Pr(p)(p - v)$, given a seller cost (or buyer value) of v . Buyers utilize a symmetric heuristic.

The original GD was further improved (e.g., MGD by Tesouro and Das [28]) and tested by several authors, who confirmed GD to be relatively strong. In particular, Walsh et al. [33] conducted an empirical game-theoretic analysis including ZIP, GD, and Kaplan and found that GD would become nearly dominant with only a 5% improvement. Our implementation mimics MGD, using $M = 7$.

3.5 GDX

GD attempts to optimize surplus, but in a myopic fashion, considering only the next immediate trade. To address this limitation, Tesouro and Bredin [27] proposed an online dynamic programming (DP) bidding approach for the CDA, dubbed *GDX*. This strategy represents a state by the agent’s pending trades and remaining bidding opportunities, and estimates transition probabilities using Gjerstad and Dickhaut’s belief function. The approach maximizes discounted cumulative future rewards rather than just immediate profits. GDX weights rewards using a discount parameter γ , and the authors note that as $\gamma \rightarrow 0$ GDX reproduces the GD strategy, whereas with $\gamma \rightarrow 1$ GDX deviates the most from GD.

In order to determine a new bidding price, GDX recalculates its DP recursion on every bidding iteration. The authors found experimentally that high values of γ create a strong GDX trader that clearly outperforms GD in a wide variety of market scenarios. Our implementation uses $\gamma = 0.9$.

3.6 Risk-Based and Adaptive-Aggressiveness

Vytelingum et al. [32] proposed a “risk-based” (RB) strategy that, much like ZIP (but using a more sophisticated method), gradually adjusts expected margins based on market activity. In empirical studies with homogeneous and balanced populations of two strategies, the authors found that RB outperformed ZI and ZIP.

The strategy determines a risk factor r to classify agents as: (a) trying to get high profits with a low probability of succeeding ($0 < r \leq 1$); (b) seeking lower profits with a higher probability of trading ($-1 \leq r < 0$); or (c) neutral ($r = 0$). RB uses an arbitrary function to calculate a target price $\tau(p^*, r)$, where p^* is a moving average of historic trade prices. The value of τ is also determined by $\theta \in [-1, \infty)$, a fixed parameter that specifies the rate of change in τ with respect to r . Seller agents use a simple learning rule to increase r (which results in a larger τ) when trade occurs at a price $p \geq \tau$, and decrease it when trading at $p < \tau$ or submitting sell bids

at $p \leq \tau$. Buyers utilize a symmetric approach. On each bidding iteration, the actual bidding price is moved towards τ by a fixed factor. Our implementation sets $\theta = 1$, which the authors found performed best in heterogeneous populations.

Further improvement of RB produced a new strategy called “adaptive aggressiveness” (AA) [31]. AA uses a learning rule to adapt θ to market conditions, increasing (decreasing) the rate of change of τ given higher (lower) price volatility. The authors evaluated this strategy experimentally, and found that it outperformed ZIP and GDX in both static and dynamic environments.

3.7 Other Strategies and Related Work

The Penn-Lehman Automated Trading (PLAT) Project [12] employs a market simulator that merges bids from automated trading agents with limit order data from real-world financial exchanges. This project inspired several bidding strategies, including three approaches explored by Sherstov and Stone [23]. One was based on a very simple RL formulation. A second “trend-following” strategy predicted prices based on a linear regression, placing purchase bids for positive regression slopes and sell bids for negative ones, closing the position when the trend reversed. The third strategy, dubbed “market-making”, also used linear regression to predict trends and place bids, closing positions as soon as a small profit was achieved.

Park et al. [16] propose a bidding heuristic based on a Markov chain model of the auction process, computing state transition probabilities and payoffs for discrete bid prices and picking the one providing the highest benefit. Their approach is similar in spirit to our learning scheme, except that we explore a larger state space with no obvious way to model transitions explicitly. He et al. [10] employ fuzzy logic to develop a heuristic strategy for the CDA, using fuzzy rules and reasoning mechanisms in order to find the “best” bid given a market state.

Other researchers employ simulations with automated strategies with the purpose of evaluating global market properties. Farmer et al. [5], for example, analyze the predictive power of a model consisting of ZI agents, using data from the London Stock Exchange. Agent models from the finance literature are also relevant to this research. LeBaron [14] surveys agent-based models used in finance, some of which include settings of autonomous agents that learn through genetic algorithms.

The Trading Agent Competition, an international forum devoted to trading research, organized a yearly tournament of a game called TAC Travel between 2000 and 2006 [34]. Over time, tournament participants explored different trading approaches for CDAs (among other auctions included in the game), ranging from ad hoc hard-coded strategies [7, 29] to machine learning methods [2, 24].

4. EXPERIMENTAL SETUP

4.1 CDA Game

The CDA game investigated in this research was inspired by many previous works, including those surveyed in Table 1. Our experimental setup was designed to calibrate as closely as possible with prior studies (not completely possible given their heterogeneity), and is especially similar to that of Walsh et al. [33].

We consider a CDA game of 16 agents with fixed roles, half of them buyers and half of them sellers of up to 10 units of a commodity. Buyers are provided with unlimited access to cash, and sellers are endowed 10 units of the commodity. The type of an agent is defined by a list v_1, \dots, v_{10} , where v_i is the value obtained by a buyer (or cost of a seller) trading the i th unit, assuming that buyers (sellers) trade units by decreasing (increasing) value.

Papers	Strategies	Profiles ^a
Tesauro and Bredin [27]	ZIP, GD, GDX	D, B
Tesauro and Das [28]	Kaplan, ZI, ZIP, GD, MGD	H, D, B
Vytelingum et al. [30]	Parameterized GDs	A
Vytelingum et al. [32]	ZI, ZIP, RB	H, B
Walsh et al. [33]	Kaplan, ZIP, GD	A

Table 1: Some studies on automated CDA bidding strategies.

^aA: all profiles; B: balanced group of two strategies; D: one agent deviating from homogeneous population; H: homogeneous population.

Values v_i are independent random integers drawn from a uniform distribution between $v_{min} = 61$ and $v_{max} = 260$, sorted afterwards for trading purposes. By design, all strategies avoid trading at a loss, and thus prices in the game are restricted to the range $[v_{min}, v_{max}]$. The payoff received by a buyer trading x units at prices p_i is $\sum_{i=1}^x v_i - p_i$, and for a seller in that circumstance $\sum_{i=1}^x p_i - v_i$.

Agents trade units sequentially, one at a time, through a single-unit CDA. The auction remains open for a trading period of 180 seconds, and agents revise bids asynchronously and continually, with a one-second sleep time between bidding iterations. The clearing price is determined by the earlier of the matched bids. Unmatched bids are kept in the order book until replaced, or until the end of the trading period. Some authors [3, 28] have imposed an additional rule known in the literature as the “NYSE convention”, requiring new bids to beat the current price quote in order to speed-up convergence towards trading prices. We do not enforce any such bid improvement rule.

Given that some strategies adapt online over time, conditioning the agent behavior on experience from previous history, each trading period is replayed five times with identical roles, unit valuations, and initial holdings (i.e., no trades). The mean payoff of these five periods constitutes a single game instance (or experiment). Note that the agents we evaluate differ significantly in the extent to which they retain information across trading periods. Specifically, Kaplan remembers extreme trade prices from the previous period, ZIP maintains margins for each unit (initializes new margins randomly to request at least what they had requested in the previous period), and GD and GDX maintain the extreme prices and the state underlying their belief function. The remaining strategies (ZI, ZIbtq, RB, AA, and all of our learned strategies) treat each trading period independently.

4.2 Reductions

Given the prohibitive computational expense of exhaustive analysis, we employ several techniques for reducing the number of simulations required.

Exploiting Symmetry.

In a symmetric game, all agents have the same strategy set, and the payoff for a strategy depends on the strategies employed by the others, but not on who is playing which.¹ By exploiting game symmetry we decrease the number of distinct strategy profiles from S^n to $\binom{n+S-1}{n}$, given n players and S strategies. For $n = 16$ and $S = 14$ (the size of our ultimate CDA experiment), symmetry reduces the profile space from over 10^{18} to about 68 million.

Hierarchical Reduction of Players.

Hierarchical reduction [36] restricts the number of agents playing any strategy to a multiple of some fixed integer, coarsening the profile space by reducing the degrees of strategic freedom. Assum-

¹The CDA game distinguishes buyer and seller roles, but since the behaviors and payoffs are mirrored we can still treat as symmetric.

ing that payoffs are not too sensitive to unilateral strategy deviations and vary smoothly with the quantity of agents playing them, a reduced game can serve as a good approximation of its underlying full game. We employ this reduction to transform the original 16-player into a 4-player CDA game, denoted by $CDA_{\downarrow 4}$. For 14 strategies, the combined effect of symmetry and hierarchical reduction shrinks the space to 2380 profiles.

Variance Reduction.

The method of control variates [21] is a standard technique to reduce variance in Monte Carlo simulation. The idea is to exploit the correlation between agent type and payoff, adjusting payoffs based on some measure of “luck”. Given an agent type, the method calculates the expected payoff and subtracts it from sampled payoff, reducing variance while preserving the mean. To apply control variates, we estimate expected payoff by using a linear regression fitting sampled payoffs of a buyer as a function of normalized unit valuations, $v_i - v_{min}$ (for sellers, the normalized valuations are $v_{max} - v_i$). The dataset used to calculate coefficients included 25,000 games picked randomly from profiles of strategies 1 to 6 (see Table 3), each game involving 80 points (one per agent and trading period). All results reported here are based on this adjustment, which resulted in a variance reduction of 62%.

4.3 Implementation

We implemented a CDA game simulator using *AB3D*, a configurable market game server [15]. Our distributed setup simulated many games in parallel, using almost 200 Linux machines at the University of Michigan. Each game runs on two separate machines, one controlling the game and operating the auction, and another running all 16 agents, each a separate process. Our learning infrastructure (Section 6) was also distributed, comprising on average 30 clients running the online learning algorithm and updating a centralized database in parallel.

5. METHODOLOGY

The basic approach to EGTA involves writing a game simulator, generating candidate strategies, estimating an empirical payoff matrix, and analyzing it using standard tools from game theory. In order to generate candidate strategies, researchers often construct a baseline policy informed by their domain knowledge, and explore a strategy space defined by parametric variations on the baseline. The key extension we introduce is to generate candidate strategies for our CDA game automatically, through reinforcement learning. We employ Q-learning, a classic model-free RL approach, which is appropriate given our limited basis for modeling the dynamic behavior of the CDA environment.

Our methodology consists of the following broad steps:

1. Implement a game simulator that returns payoffs as a function of agent strategies, randomly chosen agent types, and other sources of randomness intrinsic to the game. (Section 4)
2. Implement a set of candidate strategies S . (Section 3)
3. Estimate the empirical game via Monte Carlo sampling, by running the simulator repeatedly. (Section 7.2)
4. Find a Nash equilibrium s^* . (Section 7.2)
5. Derive a new bidding strategy L using RL, applied in a context where other agents play s^* . (Sections 6 and 7.1)
6. If L provides a positive deviation from s^* , add L to S , and extend the empirical game by continuing with step 3. Otherwise, if learning has converged and the RL model cannot be improved further, the process ends.

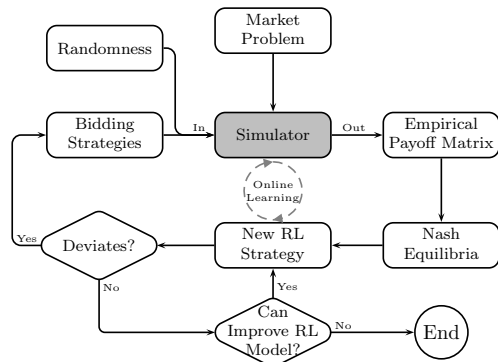


Figure 1: Interleaving EGTA and RL.

Steps 1 through 4 are part of the standard EGTA process [20]. Our extension adds steps 5 and 6, and repeats the cycle until convergence. The entire process is summarized in Figure 1.

The central idea in our interleaved approach is to focus learning effort on the contexts (i.e., configuration of other-agent strategies) supported by equilibrium reasoning over the data collected thus far. Given the large amount of training data required for effective RL in this domain, it would not be practical to learn a best response to any but a tiny fraction of other-agent strategy profiles. By focusing on finding a deviation from s^* , we concentrate the training on the most promising regions of profile space. By definition, a new strategy that succeeds at deviating from s^* will qualitatively change the empirical game analysis, effectively producing a new equilibrium. By introduction of relevant strategies at successive EGTA-RL iterations, we increase our confidence in the ultimate results of analyzing the cumulative empirical game.

Throughout this work we utilize the standard notion of approximate equilibrium, measured by the *regret* value ϵ . A strategy profile s constitutes an ϵ -equilibrium if no player could expect to gain more than ϵ by deviating unilaterally from its strategy in s to any feasible strategy [19]. The regret of a Nash equilibrium (NE) is zero. There are numerous methods to compute Nash equilibrium given a payoff matrix. Our choice is to employ replicator dynamics [26], an evolutionary method that iteratively updates the proportions of a population of pure strategies. When the process converges, the result of replicator dynamics is a symmetric equilibrium (in general, a mixed strategy with probabilities defined by the proportions).

6. LEARNING FRAMEWORK

6.1 Reinforcement Learning Model

Reinforcement learning [25] is a machine learning technique that aims to discover a policy that maximizes reward over time. In contrast to supervised learning, in which an agent is taught by example, RL requires discovering what actions produce the best results by trying them. In order to derive bidding strategies for the CDA, we employ a variety of RL called Q-learning, and define our model by a standard formulation of states, actions, and rewards.

State Space.

We consider the following candidate features to describe a state. These features define the observable variables eligible for conditioning actions.

- H_1 Normalized moving average \bar{p} of recent trade prices from all agents. For buyers, $H_1 = \bar{p} - v_{min}$; for sellers, $H_1 = v_{max} - \bar{p}$.
- H_2 Probability weighted ratio of recent trade prices from all agents

above and below a threshold d . For a seller,

$$H_2 = \frac{\int_d^{v_{max}} [1 - F(p)] dp}{\int_{v_{min}}^d F(p) dp},$$

where $F(p)$ is the cumulative distribution of recent trade prices, and d is the value of the next unit to be traded. Buyers use a symmetric formula. This feature is similar to the Omega measure used in finance [13].

H_3 Same as H_2 , but using the bid price as a threshold. This feature is actually a hybrid between a state and an action. It is related to the belief function used by GD, as it embeds in a state representation an implicit likelihood of trading, based on recent price history.

Q_1 Best outstanding bid by agents of the opposite role. For buyers, $Q_1 = ASK - v_{min}$; for sellers, $Q_1 = v_{max} - BID$.

Q_2 Best outstanding bid by agents of the same role. For buyers, $Q_2 = BID - v_{min}$; for sellers, $Q_2 = v_{max} - ASK$.

T_1 Total time elapsed this trading period.

T_2 Time since the last trade.

U Number of units left to trade.

V Normalized value v_i of the next unit to be traded. For buyers, $V = v_i - v_{min}$; for sellers, $V = v_{max} - v_i$.

Actions.

An action A is defined as a positive offset from the value of the next unit to be traded, which determines the bidding price.

Rewards.

Agents receive as immediate reward the surplus for each completed trade, that is, the difference between unit valuation and trading price. Terminal rewards are not needed in our CDA game.

6.2 Function Approximation with Tile-Coding

A Q function can be represented in tabular or implicit form. Many RL applications (like our CDA scenario) involve continuous features and high dimensionality, requiring function approximation. Several successful function approximators exist. Here we use a (mostly) standard implementation of tile-coding [25], a computationally efficient method that provides local generalization.

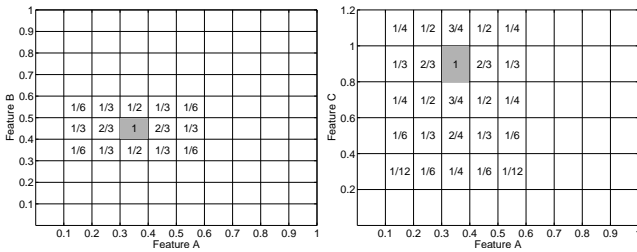


Figure 2: Update proportions for two tilings encoding features (A, B, C) = (.33, .48, .91), given $b_A = 2$, $b_B = 1$, and $b_C = 3$.

We partition state and action features into tiles, and combine them into one or many multidimensional tilings. The method maintains a weight on each tile, and the approximate Q-value of a state-action pair is represented by the sum of the weights of the tiles, one per tiling, in which it is contained. To generalize across multiple dimensions (features), we define a parameter b_i to denote *generalization breadth*, the farthest neighbor tile across dimension i that gets updated. Given a training tuple, the method finds the containing tile t (in each tiling), calculates the standard Q-learning update, and adjusts the weight on t accordingly. Neighbors that are

d_i ($d_i \leq b_i$) tiles away from t across dimension i get a fraction of such update equal to $\prod_{i=1}^F (1 - \frac{d_i}{b_i+1})$, assuming F features encoded in the tiling. An example is provided in Figure 2. Note that multidimensional updates grow exponentially in the number of dimensions, but are manageable in practice given small b_i s and relatively few dimensions per tiling.

7. EXPERIMENTS

7.1 Learned Strategies

Even our small set of candidate features induces too many possible tiling arrangements to test them all.² We tried combinations of features based on our experience and some experimentation, over 14 learning iterations described in Table 2.

Strategy	State	Action
L1	$Q_1^{(1,10)}$ $Q_2^{(1,10)}$ $T_1^{(1,8)}$ $U^{(0,10)}$ $V^{(1,10)}$	$A^{(2,10)}$
L2	$Q_1^{(1,10)}$ $Q_2^{(1,10)}$ $T_1^{(1,8)}$ $U^{(0,10)}$ $V^{(1,10)}$	$A^{(2,10)}$
L3	$Q_1^{(1,16)}$ $Q_2^{(1,16)}$ $T_1^{(1,8)}$ $U^{(0,10)}$ $V^{(1,16)}$	$A^{(0,16)}$
L4	$Q_1^{(1,16)}$ $T_1^{(1,8)}$ $U^{(0,10)}$ $V^{(1,16)}$ $Q_2^{(1,16)}$ $T_1^{(1,8)}$ $U^{(0,10)}$ $V^{(1,16)}$	$A^{(0,16)}$ $A^{(0,16)}$
L5	$Q_1^{(1,16)}$ $T_2^{(1,8)}$ $U^{(0,10)}$ $V^{(1,16)}$	$A^{(0,16)}$
L6	$Q_1^{(1,16)}$ $T_2^{(1,8)}$ $U^{(0,10)}$ $V^{(1,16)}$ $Q_2^{(1,16)}$ $T_2^{(1,8)}$ $U^{(0,10)}$ $V^{(1,16)}$	$A^{(0,16)}$ $A^{(0,16)}$
L7	$H_1^{(1,16)}$ $Q_1^{(1,10)}$ $T_2^{(1,8)}$ $U^{(0,10)}$ $V^{(1,8)}$	$A^{(0,16)}$
L8	$H_2^{(1,16)}$ $Q_1^{(1,10)}$ $T_2^{(1,8)}$ $U^{(0,10)}$ $V^{(1,8)}$	$A^{(0,16)}$
L9-L14	$H_3^{(1,16)}$ $Q_1^{(1,10)}$ $T_2^{(1,8)}$ $U^{(0,10)}$ $V^{(1,8)}$	$A^{(0,16)}$

Table 2: Features encoded into strategies L1–L14. Feature i is denoted by $i^{(b_i, r_i)}$, for generalization breadth b_i and r_i tile partitions. Strategies on multiple rows denote multiple tilings.

Strategy L1 was learned offline in two successive learning iterations, each using a different training set. The first iteration used games in which all agents played ZI. The second used games with 4 learning agents (derived from the first iteration) attempting to deviate from an all-ZI population. We always use a 4-agent deviation, equivalent to one player in our reduction to CDA₄.

All subsequent strategies, L2 to L14, were derived using on-line Q-learning. A training set consisted of games in which 12 agents played according to an equilibrium mix, and four learning agents attempted to deviate. Consequently, the proportion of training games in each profile was consistent with that mix. Training was conducted by repeatedly cycling over the experience collected by all 16 agents during the last 250 games played, out of 7000 games. Each training tuple was used 1407 times, on average. During the first 3000 games, learning agents explored new actions with a 15% probability, using softmax action selection. Thereafter, learning agents always picked the best action for 4000 additional games. In all cases, the learning rate was fixed at .01, and the discount factor at .99. The payoff of a learned strategy was evaluated by playing all successive games with no further adaptation.

7.2 Interleaving EGTA and RL

We fully explored CDA₄ by running our distributed testbed (almost continuously) for several months, sampling at least 100 games per profile. The results of our experiments are summarized in Table 3. Figure 3 shows the learning curves of strategies L9 to L14.

²Given $1 \leq F \leq 9$ and one tiling, there are $2^9 - 1$ possible tiling arrangements, and even more when considering multiple tilings or all possible values of b_i . The possibilities are infinite when considering all possible ways to split features into tiles.

EGTA				Learning	
Strategies	Equil. Mix	Payoff	Num. Profiles	Strat.	Dev. Payoff
1. Kaplan					
2. ZI					
3. ZIbtq	ZI 1.000	248.1	15	L1	268.7
4. L1	L1 1.000	242.5	35		
5. ZIP	ZIP 1.000	248.0	70		
6. GD	GD 1.000	248.6	126	L2 L3 L4 L5 L6 L7 L8 L9	228.8 225.0 228.2 233.0 229.9 237.0 237.6 251.8
7. L9	GD .531 L9 .469	246.1	210	L10	252.1
8. L10	GD .191 L10 .809	248.0	330	L11	251.0
9. L11	L11 1.000	246.2	495		
10. GDX	GDX .192 L11 .808	245.8	715	L12	248.3
11. L12	L11 .049 L12 .951	245.8	1001	L13	245.9
12. L13	L12 .872 L13 .128	245.6	1365	L14	245.6
13. RB	L12 .872 L13 .128	245.6	1820		
14. AA	L12 .872 L13 .128	245.6	2380		

Table 3: Results obtained by interleaving EGTA with RL.

We first implemented Kaplan, ZI, and ZIbtq, and identified an all-ZI Nash equilibrium. Our initial learned strategy, L1, produced a positive deviation, and a new NE with all agents playing L1. Then, we implemented stronger contenders, namely ZIP and GD, and found a new NE with all agents playing GD. The following attempts to deviate, from L2 to L8, were unsuccessful, until we tried strategy L9 which did produce a positive deviation and a new mixed-strategy NE with GD (.531) and L9 (.469). The next deviation attempt, with strategy L10, was successful as well, and produced a new mixed-strategy NE with GD (.191) and L10 (.809). Finally, L11 produced yet another deviation, and a pure-strategy NE with all agents using L11.

We then tried an even stronger contender, GDX, which resulted in a new equilibrium with GDX (.192) and L11 (.808). The next two iterations produced deviations as well and new equilibria, first L11 (.049) and L12 (.951), and then L12 (.872) and L13 (.128). The next attempt to deviate with L14, however, failed, and equilibrium remained the same. Lastly, we tried strategies RB and AA, which did not change the equilibrium previously found. Our methodology had converged into a new L12/L13 equilibrium, which did not change with further learning, defeating all other strategies that we tested. Further search for Nash equilibria via function minimization revealed 16 mixtures with regret less than 0.01, all combinations of L11 (0-11%), L12 (82-90%), and L13 (0-18%) only. Other RL encodings could possibly provide a better performance, but we leave further exploration to follow-on investigations. It is interesting to note the mostly decreasing margin of deviation achieved with every iteration, from 20.6 points (L1), to 0.1 points (L13), and finally no deviation (L14).

A desirable property of our methodology (or any other incremental discovery of strategies) is that new strategies provide a monotonically decreasing regret (epsilon) with respect to the “true” equilibrium (i.e., with respect to the entire strategy space). Of course,

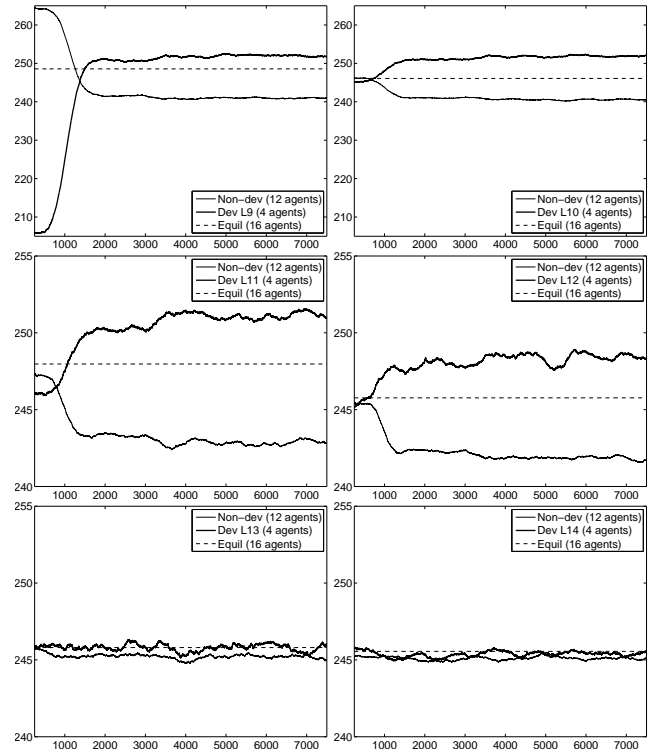


Figure 3: Learning curves of L9-L14 (4 agents attempting to deviate from equilibrium), payoffs of 12 non-deviating agents, and equilibrium payoffs from Table 3. Strategies L9-L12 resulted in a clear deviation, L13 deviated only slightly, and L14 was unable to deviate. All curves show moving averages with a window of 500 games. Games beyond 7000 show final performance without further adaptation.

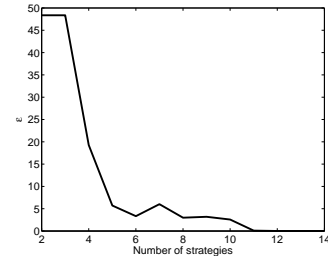


Figure 4: Deviations from each incremental equilibrium shown in Table 3 to strategies 1-13.

we cannot provide any such guarantee, nor compute that regret (if we knew the theoretical equilibrium, then our methodology would not be needed). One thing we can do, as shown in Figure 4, is identify retroactively the best possible deviation to any strategy tested from each incremental equilibrium. The mostly decreasing curve is encouraging in the sense that incremental refinements of equilibrium seem to be moving regret in the right direction. Note that, by definition, the last point of the curve will always be zero.

7.3 Analysis of L12 and L13

Throughout this research, we treated learned strategies as “black boxes” encoded in large tile-coding databases and defined by their underlying RL models. To better understand these strategies, we use regression trees to characterize their behavior as a function of

observable features. We say that an action A_1 (as defined in Section 6.1) *requests more margin* than another A_2 if it seeks a larger immediate profit (i.e., $A_1 - A_2 > 0$).

Our trees fit the differences between the actions of any two strategies (response variable) as a function of observed state features (predictors). We run 1,000 games (16,000 when considering all agents) with everyone playing the equilibrium strategy consisting of L12 (.872) and L13 (.128). For each game, we save the state and actions that GDY and L12 would take at every bidding opportunity, a total of $1.3e+06$ state-action observations per strategy.

Our predictors include all state features described in Section 6.1 (except H_3 which co-varies with the action), along with the following additional features:

H_{1b} For buyers, $H_{1b} = \bar{p} - v_i$; for sellers, $H_{1b} = v_i - \bar{p}$.

H_2^0 Binary feature equal to 1 in case $H_2 = 0$, or 0 otherwise.

H_2^∞ Binary feature equal to 1 in case $H_2 = \infty$, or 0 otherwise.

Q_3 BID-ASK spread.

Y_1 Binary feature equal to 1 if no bids have been submitted to the auction (by any agent), 0 otherwise.

Y_2 Binary feature equal to 1 if no trades have yet occurred, 0 otherwise.

To compare any two strategies, we use the observations from 500 games to build the largest possible tree t that splits nodes until they are either pure (all observations provide the same response), contain identical predictor values, or contain less than 250 observations. We then use the observations from the remaining 500 games to compute the cost³ of all subtrees in the optimal pruning sequence for t , and prune t to the smallest tree that is within one standard error of the minimum-cost subtree.

The comparison between L12 and GDY (actions by L12 minus actions by GDY) is shown in Figure 5. Half of the splits are based on H_{1b} (nodes 3, 7, 9, 11, 12, 13, 14, 15, 17, 23, 30, 36, 41), showing that L12 requests relatively less margin for lower values of H_{1b} . Since a lower H_{1b} is usually consistent with better prices, these splits suggest that L12 tries to take advantage of good (price) opportunities before GDY does. Consistently, splits on Q_1 (nodes 2, 6, 20, 26, 27) and H_1 (nodes 16, 22, 34) show that L12 requests relatively less margin given better quote (lower Q_1) and better trade (lower H_1) prices. The first split shows that L12 has a lower margin request when $H_2 = 0$ (low probability of trading), and as time passes (nodes 4, 5, 24). On average, L12 requests 5.2 fewer points of margin than GDY.

Space considerations preclude other strategy comparisons here. An extended version of this paper will include further comparisons.

8. DISCUSSION

We proposed a methodology for deriving bidding strategies, and applied it to a widely known CDA game. The strategies that we obtained supported a new mixed-strategy Nash equilibrium, surpassing in stability plausible versions of all other strategies published to date. Our approach can be viewed as an attempt to automate the process of the prior literature in this domain (and others as well), where a series of new strategy proposals (one per publication) is shown to improve over a selected set of prior candidates. Although the result here also takes this form, we emphasize the iterative process by which new ideas can subsequently be incorporated and improved upon. Previous efforts have also automated strategy generation to some extent using genetic search methods

³The cost of a tree is defined as the probability weighted average cost of all leaf nodes. The cost of a node is the mean squared error of the observations in that node.

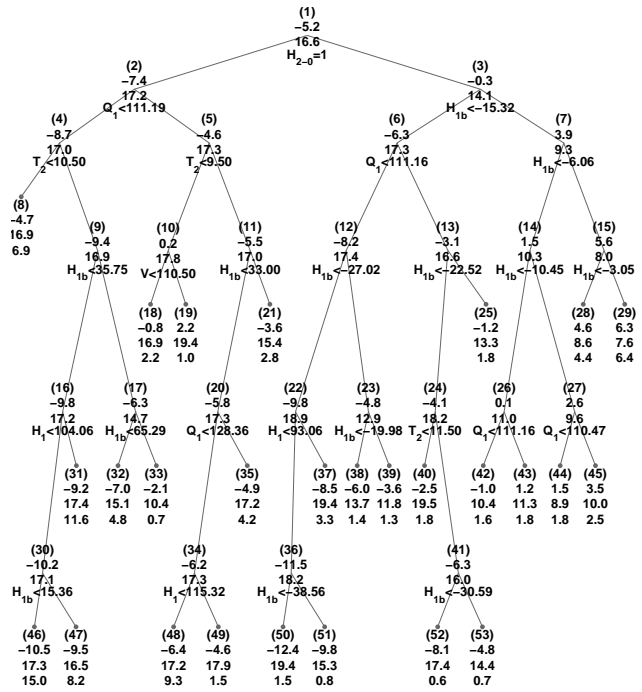


Figure 5: Regression tree comparing strategies L12 and GDY. Nodes display: (1) ID; (2) mean response difference; (3) standard deviation; (4) rule pointing to left child (choice nodes), or percentage of observations covered (leaves).

[4, 17]. Our use of RL for this function can be viewed as just an alternative optimization method, which we found effective for exploring a large non-parametric strategy space. Perhaps the more crucial distinguishing feature of our work is the appeal to game-theoretic reasoning to establish the context within which strategy optimization takes place.

An important limitation of our approach is that, given a market problem, it can be difficult to define a good RL model that actually works. Interesting games typically involve multiple agents in partially observable, non-stationary, and high-dimensional environments, which makes the learning problem challenging. As documented in Table 3 (in particular row 6), our effort here entailed significant trial-and-error. We provide no recipe for this issue beyond relying on domain knowledge, creativity, and plenty of patience. The RL models described here can serve as a good starting point, especially for games involving double auctions.

Another challenge is the time required to sample a very large payoff matrix, and search through a huge strategy space. Since many critical tasks in our methodology can run in parallel, this problem can be addressed reasonably well given sufficient time and resources. Approximations and reductions also provide essential help, while the iterative process of deviating from equilibrium focuses on limited but promising regions of the strategy space. Further research on empirical game modeling and managing the search process will enable yet more comprehensive strategic analysis of rich multiagent scenarios.

9. REFERENCES

[1] K. Cai, J. Niu, and S. Parsons. Using evolutionary game-theory to analyse the performance of trading strategies in a continuous double auction market. In *Adaptive Agents and Multi-Agents Systems*, volume 4865 of *Lecture Notes in Computer Science*, pages 44–59. Springer-Verlag, 2007.

- [2] S.-F. Cheng, E. Leung, K. M. Lochner, K. O'Malley, D. M. Reeves, L. J. Schwartzman, and M. P. Wellman. Walverine: a Walrasian trading agent. *Decision Support Systems*, 39(2):169–184, 2005.
- [3] D. Cliff. Minimal-intelligence agents for bargaining behaviours in market-based environments. Technical Report HP-97-91, HP Laboratories, 1997.
- [4] D. Cliff. ZIP60: Further explorations in the evolutionary design of online auction market mechanisms. Technical Report HPL-2005-85, HP Laboratories, 2005.
- [5] J. D. Farmer, P. Patelli, and I. I. Zovko. The predictive power of zero intelligence in financial markets. *Proceedings of the National Academy of Sciences*, 102:2254–2259, 2005.
- [6] D. Friedman and J. Rust, editors. *The Double Auction Market: Institutions, Theories, and Evidence*. Addison-Wesley, 1993.
- [7] C. Fritschi and K. Dorer. Agent-oriented software engineering for successful TAC participation. In *First International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 45–46, Bologna, Italy, 2002.
- [8] S. Gjerstad and J. Dickhaut. Price formation in double auctions. *Games and Economic Behavior*, 22(1):1–29, 1998.
- [9] D. K. Gode and S. Sunder. Allocative efficiency of markets with zero intelligence traders: Market as a partial substitute for individual rationality. *Journal of Political Economy*, 101(1):119–137, 1993.
- [10] M. He, H. fung Leung, and N. R. Jennings. A fuzzy logic based bidding strategy for autonomous agents in continuous double auctions. *IEEE Transactions on Knowledge and Data Engineering*, 15(6):1345–1363, 2003.
- [11] P. R. Jordan, C. Kiekintveld, and M. P. Wellman. Empirical game-theoretic analysis of the TAC supply chain game. In *Sixth International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1188–1195, 2007.
- [12] M. Kearns and L. Ortiz. The Penn-Lehman automated trading project. *IEEE Intelligent Systems*, 18(6):22–31, 2003.
- [13] C. Keating and W. Shadwick. *Omega: Functions and Metrics*. Gilmour Drummond Publishing, 2005.
- [14] B. LeBaron. Agent-based computational finance. In L. Tesfatsion and K. L. Judd, editors, *Handbook of Computational Economics: Agent-Based Computational Economics*, volume 2. North-Holland, 2006.
- [15] K. M. Lochner and M. P. Wellman. Rule-based specification of auction mechanisms. In *Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 818–825, New York, 2004.
- [16] S. Park, E. H. Durfee, and W. P. Birmingham. Use of Markov chains to design an agent bidding strategy for continuous double auctions. *Journal of Artificial Intelligence Research*, 22:175–214, 2004.
- [17] S. Phelps, M. Marcinkiewicz, S. Parsons, and P. McBurney. A novel method for automatic strategy acquisition in n -player non-zero-sum games. In *Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 705–712, Hakodate, 2006.
- [18] C. Preist. Commodity trading using an agent-based iterated double auction. In *Third International Conference on Autonomous Agents*, pages 131–138, 1999.
- [19] R. Radner. Collusive behavior in noncooperative epsilon-equilibria of oligopolies with long but finite lives. *Journal of Economic Theory*, 22(2):136–154, 1980.
- [20] D. M. Reeves. *Generating Trading Agent Strategies: Analytic and Empirical Methods for Infinite and Large Games*. PhD thesis, University of Michigan, 2005.
- [21] S. M. Ross. *Simulation*. Academic Press, third edition, 2001.
- [22] J. Rust, J. H. Miller, and R. G. Palmer. Behavior of trading automata in a computerized double auction market. In Friedman and Rust [6], pages 155–198.
- [23] A. Sherstov and P. Stone. Three automated stock-trading agents: A comparative study. In *AAMAS-04 Workshop on Agent-Mediated Electronic Commerce*, pages 173–187, 2004.
- [24] P. Stone, M. L. Littman, S. Singh, and M. Kearns. ATTac-2000: An adaptive autonomous bidding agent. *Journal of Artificial Intelligence Research*, 15:189–206, 2001.
- [25] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.
- [26] P. D. Taylor and L. B. Jonker. Evolutionary stable strategies and game dynamics. *Mathematical Biosciences*, 40:145–156, 1978.
- [27] G. Tesauro and J. L. Bredin. Strategic sequential bidding in auctions using dynamic programming. In *First International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 591–598, 2002.
- [28] G. Tesauro and R. Das. High-performance bidding agents for the continuous double auction. In *Third ACM Conference on Electronic Commerce*, pages 206–209, 2001.
- [29] I. A. Vetsikas and B. Selman. A principled study of the design tradeoffs for autonomous trading agents. In *Second International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 473–480, 2003.
- [30] P. Vytelingum, D. Cliff, and N. R. Jennings. Evolutionary stability of behavioural types in the continuous double auction. In *AAMAS-06 Workshop on Trading Agent Design and Analysis and Agent Mediated Electronic Commerce*, 2006.
- [31] P. Vytelingum, D. Cliff, and N. R. Jennings. Strategic bidding in continuous double auctions. *Artificial Intelligence*, 172:1700–1729, 2008.
- [32] P. Vytelingum, R. K. Dash, E. David, and N. R. Jennings. A risk-based bidding strategy for continuous double auctions. In *Sixteenth European Conference on Artificial Intelligence*, pages 79–83, Valencia, Spain, 2004.
- [33] W. E. Walsh, R. Das, G. Tesauro, and J. O. Kephart. Analyzing complex strategic interactions in multi-agent systems. In *AAAI-02 Workshop on Game-Theoretic and Decision-Theoretic Agents*, 2002.
- [34] M. P. Wellman, A. Greenwald, and P. Stone. *Autonomous Bidding Agents: Strategies and Lessons from the Trading Agent Competition*. MIT Press, 2007.
- [35] M. P. Wellman, A. Osepayshvili, J. K. MacKie-Mason, and D. M. Reeves. Bidding strategies for simultaneous ascending auctions. *B. E. Journal of Theoretical Economics (Topics)*, 8(1), 2008.
- [36] M. P. Wellman, D. M. Reeves, K. M. Lochner, S.-F. Cheng, and R. Suri. Approximate strategic reasoning through hierarchical reduction of large symmetric games. In *Twentieth National Conference on Artificial Intelligence*, pages 502–508, Pittsburgh, 2005.