A Novel Method for Strategy Acquisition and its Application to a Double-Auction Market Game

Steve Phelps
Centre for Computational Finance and Economic Agents
University of Essex
United Kingdom
sphelps@essex.ac.uk

Peter McBurney Department of Computer Science,
University of Liverpool,
United Kingdom
mcburney@liverpool.ac.uk

Simon Parsons
Department of Computer and Information Science,
Brooklyn College, City University of New York,
USA parsons@sci.brooklyn.cuny.edu.

December 18, 2008

Abstract

We introduce a novel method for strategy-acquisition in non-zero-sum n-player games, and empirically validate it by applying it to a well-known benchmark problem in this domain, viz the double-auction market. Many existing approaches to strategy-acquisition focus on attempting to find strategies that are robust in the sense that they are good all-round performers against all-comers. We argue that in many economic and multi-agent scenarios the robustness criterion is inappropriate; in contrast, our method focusses on searching for strategies that are *likely to be adopted* by participating agents. We conclude by discussing several potential applications of our algorithm, including the mechanism design problem from economics.

1 Introduction

Much work in the design of multi-agent systems (MAS) [35] has focused on the design and engineering of individual agents; for example, the problems of designing and implementing effective trading strategies for agents participating in e-commerce market places, or the design of effective learning algorithms for adaptive agents. However, increasingly attention is being turned to the design of the infrastructure, or the environment, underlying the interactions between individual agents in a MAS; for example, the problem of designing rules governing the operation of an e-commerce market institution, or the design of interaction protocols governing agent argumentation. The justification for the latter approach is that often as MAS designers we are responsible for engineering *open* systems, in which we do not have control over the exact behavior of the agents connecting to our system; these agents are, after all, autonomous. Rather, we build a set of standards and protocols with which our agents are free to interact, and if we have designed our infrastructure robustly, the system as a whole will exhibit our desired design properties despite the fact that it consists of possibly millions of autonomous agents interacting with each other in ways we have not prescribed in advance.

Economists have studied similar design problems in the context of *auction theory* [13] and *mechanism design* [28]. In a mechanism design problem, the task of the designer is to choose the rules of a game, such as an auction, in such a way that the designer's objectives are met when agents play their optimal strategies. Thus mechanism design has become an important foundation of distributed Artificial Intelligence, as summarised by Wellman:

> "*Within economics, the problem of synthesizing an interaction protocol via which rational agents achieve a socially desirable end is called mechanism design. This is exactly the problem we face in designing distributed software systems, which suggests an opportunity to exploit existing economic ideas.* " [34]

One of the main difficulties in solving such problems is computing the optimal strategies, as the best strategy to play depends on what strategies are being played by other agents; the number of agents can vary significantly, and the strategy space can be very large. The standard technique is to view each possible set of auction rules as defining a particular game, and then to use game theory to "solve" this game by finding the set of strategies comprising a *Nash equilibrium* of the game [17]. Once we know the equilibrium behavior of each agent in the system, we can then compute system-level design properties such as the overall market efficiency. This approach has been very successful when applied to the restricted space of mechanisms considered by auction theory, but suffers from several drawbacks when we attempt to apply it to more complex mechanism-design scenarios.

Firstly, the game-theoretic approach assumes that there are a small number of apriori-known strategies for each player of the game, which can be structured into a normal-form payoff matrix. However, in any realistic multi-agent interaction, the space of possible actions for each agent can be extremely large (and may not be fully known) making computation of the Nash equilibria intractable in the general case [4].

Secondly, in the general case we may observe multiple equilibria for a given game and it may not be clear which of these multiple potential outcomes are *likely* to be adopted in the long-run, or indeed how long the system will take to equilibrate. Traditional game theory is a static analysis and does not give any insight into the *dynamics* of adjustment to equilibrium, making it extremely difficult to gain meaningful insights into equilibrium selection, or off-equilibrium behavior.

These difficulties have led to researchers to use heuristic methods, such as evolutionary computing, to sample the strategy space to study: i) how the behaviour an ensemble of agents is likely to converge in any given setting; and ii) to search for new, previously unconsidered strategies for the game at hand. As we shall see both of these problems are strongly interrelated.

The most promising technique from evolutionary computing for discovering new strategies is *co*-evolution, in which the fitness of each individual in an evolving population[1] of strategies is assessed relative to other individuals by computing the payoffs obtained when the selected individuals interact [12]. Co-evolution can sometimes result in *arms-races*, in which the complexity and robustness of strategies in the population increases as they counter-adapt to adaptations in their opponents.

Often, however, co-evolutionary learning can fail to converge on robust strategies. In this paper we explore some of the limitations of current co-evolutionary algorithms, and review a field known as *empirical game theory* which combines game-theoretic analysis together with simulation methods in order to analyse the strategic interaction amongst an *existing* set of strategies. We then introduce a novel technique based on empirical game-theory that is able to acquire *new* strategies for the game at hand.

This paper focuses on a specific problem domain – the double auction. The double auction has come to be recognized as an important *benchmark problem,* in both economics and multi-agent systems. In particular, a landmark workshop held in Santa Fe [25] motivated much contemporary research in this area by highlighting the difficulty of agents' decision problems in non-idealized variants of this type of marketplace, and the Santa Fe double-auction tournament was one of the first studies which used advanced agent-based simulation in order to explore the properties of a realistic economic mechanism [25]. To this day the double-auction represents an important benchmark problem by simultaneously admitting of precise representations whilst stretching the bounds of both analytical tractability and computational brute-force; for example equilibria are only known for restricted versions of the mechanism [37].

The outline of this paper is as follows. In section 2 we give an overview of game-theory and discuss the use of co-evolutionary algorithms and empirical game-theory to search for approximations of game-theoretic equilibria. In section 3 we describe in detail the specific problem domain we will attack, viz. the double-auction. In section 4 we describe the search space of strategies for this game, and in section 5 we use empirical game-theory to analyse the strategic-interaction between existing strategies within this space with a view to identifying potential candidates for optimisation. In section 6 we describe a novel method for strategy acquisition, and in section 7 we present the results of an empirical validation of our technique. In section 8 we discuss generalizations and applications of our approach, and finally we conclude with a discussion of strengths and weaknesses in section 9.

[1]Or sometimes several populations.

2 Co-evolution and empirical game theory

2.1 Nash Equilibrium

The failure of certain types of co-evolutionary algorithms to converge on robust strategies in certain scenarios is well known [2, 7, 32], and has many possible causes; for example, the population may enter a limit cycle if strategies learnt in earlier generations are able to exploit current opponents and current opponents have "forgotten" how to beat the revived living fossil. Whilst many effective techniques have been developed to overcome these problems, there remains, however, a deeper problem which is only beginning to be addressed successfully. In some games, such as Chess, we can safely bet that if player A consistently beats player B, and player B consistently beats player C, then player A is likely to beat player C. Since the dominance relationship is transitive, we can build meaningful *rating systems* [26] for objectively ranking players in terms of ability, and the use of such ranking systems can be used to assess the "external" fitness of strategies evolved using a co-evolutionary process and ensure that the population is evolving toward better and better strategies. In many other games, however, the dominance graph is highly intransitive, making it impossible to rank strategies on a single scale. In such games, it makes little sense to talk about "best", or even "good", strategies since even if a given strategy beats a large number of opponent strategies there will always be many opponents that are able to beat it. The best strategy to play in such a game is always dependent on the strategies adopted by one's opponents.

Game theory provides us with a powerful concept for reasoning about the best strategy to adopt in such circumstances: the notion of a *Nash equilibrium*. A set of strategies for a given game is a Nash equilibrium if, and only if, no player can improve their payoff by unilaterally switching to an alternative strategy.

If there is no dominant strategy (a strategy which is always the best one to adopt no matter what any opponent does) for the game, then we should play the strategy that gives us the best payoff based on what we believe our opponents will play. If we assume our opponents are payoff maximisers, then we know that they will play a Nash strategy set by *reductio ad absurdum*; if they did not play Nash then by definition at least one of them could do better by changing their strategy, and hence they would not be maximising their payoff. This is very powerful concept, since although not every game has a dominant strategy, every finite game possesses at least one *equilibrium* solution [17].

Note, however, that the Nash strategy is not always the *best* strategy to play in all circumstances. In 2-player zero-sum games, the minimax theorem tells us that even if there are multiple equilibria, any equilibrium strategy is guaranteed to obtain a certain payoff known as the security-level of the game regardless of the opponent's actions. Thus in these scenarios, we have a clear metric for the *robustness* of a strategy since if a particular course of action yields less than the value of the game we can infer that we are being exploited. However, this result does not hold when we generalise to n-player non-zero-sum games; in such games, if there are multiple equilibria they may yield different payoffs to the same player, and thus the outcome is not clear-cut. Additionally, in any game constant-sum or otherwise, players may be able to obtain a better payoff than their security-level by countering a non-equilibrium strategy with another non-equilibrium strategy.

2.2 Beyond Nash equilibrium

Standard game theory does not tell us which of the many possible Nash equilibria are likely to be played. *Evolutionary* game theory [16] and its variants attack this problem by positing that, rather than computing the Nash strategies for a game using brute-force and then selecting one of these to play, our opponents are more likely to gradually adjust their strategy over time in response to repeated observations of their own and others' payoffs. One approach to evolutionary game-theory uses the *replicator dynamics* [16] equation to specify the frequency with which different pure strategies should be played depending on our opponent's strategy:

$$\dot{m}_j = [u(e_j, \vec{m}) - u(\vec{m}, \vec{m})]\, m_j \tag{1}$$

where $\vec{m}$ is a mixed-strategy vector, $u(\vec{m}, \vec{m})$ is the mean payoff when all players play strategy $\vec{m}$, $u(e_j, \vec{m})$ is the average payoff to pure strategy j when all players play $\vec{m}$, and $\dot{m}_j$ is the first derivative of m_j with respect to time. Strategies which gain above-average payoffs become more likely to be played, and this equation models a simple *co-evolutionary* process of mimicry learning, in which agents switch to strategies that appear to be more successful.

For any initial mixed-strategy we can find the eventual outcome from this co-evolutionary process by solving $\dot{m} = 0 \ \forall_j$ to find the final mixed-strategy of the converged population, i.e., the stationary points of the process. This model has the attractive properties that: (i) all Nash equilibria of the game are stationary points under the replicator dynamics; and (ii) all Lyapunov stable states [14] and interior limit states are also Nash equilibria [33, pp. 88–89][2].

Thus the Nash equilibrium solutions are embedded in the stationary points of the direction field of the dynamics specified by equation 1. Although not all stationary points are Nash equilibria, by overlaying a dynamic model of learning on the equilibria we can see which solutions are more likely to be discovered by *boundedly-rational* agents. Those Nash equilibria that are stationary points at which a larger range of initial states will end up, are equilibria that are more likely to be reached (assuming an initial distribution of strategies that is uniform).

This is all well and good in theory, but the model is of limited practical use since many interesting real-world games are *multi-state*[3]. Such games can be transformed into normal-form games, but only by introducing an intractably large number of pure strategies, making the payoff matrix impossible to compute.

2.3 Co-evolution

But what if we were to approximate the replicator dynamics by using an evolutionary search over the strategy space? Rather than considering a very large population consisting of a mixture of all possible pure strategies as per evolutionary game theory, we use

[2]It is important to note, nevertheless, that it is not the case that *all* stationary points are Nash equilibria.
[3]The payoff for a given move at any stage of the game depends on the history of play.

a small finite population of randomly sampled strategies to approximate the game. By introducing mutation and cross-over, we can search hitherto unexplored regions of the strategy space. Might such a process converge to some kind of approximation of a true Nash equilibrium? Indeed, this is one way of interpreting existing co-evolutionary algorithms; fitness-proportionate selection plays a similar role to the replicator dynamics equation in ensuring that successful strategies propagate, and genetic operators allow them to search over novel sets of strategies. There are a number of problems with such approaches from a game-theoretic perspective, however, which we shall discuss in turn.

Firstly, the proportions of the population playing different strategies serve a dual role in a co-evolutionary algorithm [9]. On the one hand, the proportion of the population playing a given strategy represents the probability of playing that pure strategy in a mixed-strategy Nash equilibrium. On the other hand, evolutionary search requires diversity in the population in order to be effective. This suggests that if we are searching for Nash equilibria involving mixed-strategies where one of the pure strategy components has a high frequency, corresponding to a co-evolutionary search where a high percentage of the population is adopting the same strategy, then we may be in danger of over-constraining our search as we approach a solution.

Secondly and relatedly, although the final set of strategies in the converged population may be best responses to each other, there is no guarantee that the final mix of strategies cannot be invaded by other yet-to-be-encountered strategies in the search space, or even by strategies that became extinct in earlier generations because they performed poorly against an earlier strategy mix that differed from the final converged strategy mix. Genetic operators such as mutation or cross-over will be poor at searching for novel strategies that could potentially invade the newly established equilibrium because of the dual role played by population frequencies. If these conditions hold, then the final mix of strategies is implausible as a true Nash equilibrium or Evolutionay Stable State (ESS), since there will be unsearched strategies that could potentially break the equilibrium by obtaining better payoffs for certain players. We might, nevertheless, be satisfied with the final mix of strategies as an approximation to a true Nash equilibrium on the grounds that if our co-evolutionary algorithm is unable to find equilibrium-breaking strategies, then no other algorithm will be able to do so. However, as discussed above, we expect *a priori* that co-evolutionary algorithms will be particularly *poor* at searching for novel strategies once they have discovered a (partial) equilibrium.

Finally, co-evolutionary algorithms employ a number of different selection methods, not all of which yield population dynamics that converge on game-theoretic equilibria [8].

These problems have led researchers in co-evolutionary computing to design new algorithms employing game-theoretic solution concepts [6]. In particular, Ficici and Pollack [9] describe a game-theoretic search technique for acquiring approximations of Nash strategies in large symmetric 2-player constant-sum games with type independent payoffs. In this paper, we address n-player non-constant-sum multi-state games with type-dependent payoffs. In such games, playing an equilibrium strategy (or an approximation thereof) does not guarantee a participant security against exploitation if there are multiple equilibria, and thus there is no clear-cut notion of a scalar robustness metric for assessing different strategies and ranking them on a single scale. In Section 6 we introduce a new metric for ranking equilibrium strategies based on their likelihood of actually being played. Meanwhile in the next section we give an overview of an existing technique for obtaining approximate versions of game-theoretic equilibria upon which our algorithm is based.

2.4 Empirical Game-Theory

Reeves *et al.* [22] and Walsh *et al.* [29] obviate many of the problems of standard co-evolutionary algorithms by restricting attention to small representative sample of "heuristic" strategies that are known to be commonly played in a given multi-state game. For many complex n-player games representative of real-world economic interactions, such as the double-auction, unsurprisingly none of the strategies commonly in use can be proven to be dominant over the others. Given the absence of a dominant strategy, it is then natural to ask if there are mixtures of these "pure" strategies that constitute game-theoretic equilibria.

For small numbers of players and heuristic strategies, we can construct a relatively small normal-form payoff matrix which is amenable to game-theoretic analysis. This *heuristic* payoff matrix is calibrated by running many iterations of the game; variations in payoffs due to different player-types (e.g., private valuations) or stochastic environmental factors (e.g., PRNG seed) are averaged over many samples of type information resulting in a single mean payoff to each player for each cell in the payoff matrix. Players' types are assumed to be drawn independently from the same distribution, and an agent's choice of strategy is assumed to be independent of its type, which allows the payoff matrix to be further compressed, since we simply need to specify the number of agents playing each strategy to determine the expected payoff to each agent. Thus for a game with j strategies, we represent entries in the heuristic payoff matrix as vectors of the form

$$\vec{p} = (p_1, \ldots, p_j)$$

where p_i specifies the number of agents who are playing the i^{th} strategy. Each entry $p \in P$ is mapped onto an outcome vector $q \in Q$ of the form

$$\vec{q} = (q_1, \ldots, q_j)$$

where q_i specifies the expected payoff to the i^{th} strategy. For a game with n agents, the number of entries in the payoff matrix is given by

$$s = \frac{(n + j - 1)!}{n!(j - 1)!} \qquad (2)$$

For small n and small j this results in payoff matrices of manageable size; for $j = 3$ and $n = 6$, 8, and 10 we have $s = 28$, 45, and 66 respectively. Although this technique is only tractable for small numbers of simultaneous players n, these are precisely the scenarios that are typically *more* difficult to analyse. Interactions amongst small numbers of agents afford more opportunity for individual agents to have a large effect on the final outcome, whereas systems with large numbers of interacting agents can be more readily modelled as a collection of homogeneous particle-like entities. The constraint on small j is more limiting; we shall return this issue in Section 8.1.

Once the payoff matrix has been computed we can subject it to a rigorous game-theoretic analysis, search for Nash equilibria solutions, and apply different models of learning and evolution, such as the replicator dynamics model, in order to analyse the dynamics of adjustment to equilibrium.

In this paper, we use the framework described above to search for a novel strategy for a specific trading game, viz: the double-auction. In the next section we describe this game in detail.

3 The double auction market

A double-auction is a generalisation of the more commonly-known *single-sided* auctions in which a single seller sells goods to multiple competing buyers (or the reverse). In a *double*-auction, as well as multiple buyers competing against each other resulting in price rises, multiple sellers of the same commodity compete against each other resulting in price falls. Institutions of this type are also known as exchanges, and are typically used to trade commodities whose valuations are subject to much uncertainty and can vary rapidly over time; for example, equity shares traded on stock exchanges.

There are many different variants of double-auction market, thus it is important to clearly describe the particular mechanism under scrutiny. In this paper we study a uniform-price clearing-house market, which we formally describe below. Note that we use formal notation merely to provide a concise and unambiguous description of our model in order to avoid any confusion, and not because we use formal methods to prove any properties of this mechanism. Readers who are already familiar with the operation of a double-sided clearing-house can skim over this section.

3.1 The auction model

In this section we give a formal description of the variant of the double-auction used in this paper. This model is adapted from [3, 10, 18, 36], and is an attempt to describe these different market scenarios within a unified model. In this model, time is represented in discrete slices $t \in \mathbb{N}$. We will follow the convention of representing the value of any time-dependent variable X at time t by subscripting with t: X_t.

The market place is populated by a finite number of *traders*, represented by the set $A = \{a_1, a_2, \ldots a_n\}$. A single commodity is traded in the market place. The commodity is traded in discrete, indivisible units.

Traders are divided into two distinct sets: *buyers*, represented by the set $B \subset A$; and *sellers*, represented by the set $S \subset A$, such that $S \cup B = A$ and $S \cap B = \emptyset$. We assume that agents are risk-neutral (utility increases linearly with increased wealth). Buyers purchase resource for consumption, and sellers produce resource for sale. Each agent a_i has a private valuation $v_i \in \mathbb{R}$ which determines the utility of a transaction in the marketplace. If a single unit is transacted at price p then buyers obtain utility $v_i - p$ whereas sellers obtain $p - v_i$.

3.1.1 Rounds

Trading in the market proceeds in *rounds*. Each round may consist of variable number of time slices. During each round, every trader in the market-place is given the opportunity to submit a *shout* to the auctioneer. During any given time-slice only one trader may place a *shout*.

3.1.2 Shouts

A shout is a commitment to buy or sell a prespecified quantity of commodity at a particular price. Shouts are divided into two sub-classes. An offer to sell is called an *ask*, and an offer to buy is called a *bid*. Shouts are represented as tuples of the form:

$$\rho = (\rho_c \in \{bid, ask, \emptyset\}, \rho_a \in A, \rho_p \in \mathbb{R}, \rho_q \in \mathbb{N}, \rho_t \in \mathbb{N}) \in P$$

where ρ_c is the class of offer, ρ_a is the trader making the offer, ρ_p is the price that the trader is willing to buy or sell at, ρ_q is the quantity of commodity that they are committed to trade, and ρ_t is the time at which the shout was submitted to the auctioneer. A buyer who submits a bid $b \in P$ is committed to buying at any price $p \leq b_p$. Similarly, a seller who submits an ask $a \in P$ is committed to selling a_q units at any price $p \geq a_p$. A trader may submit a *null shout* by setting $\rho_c = \emptyset$ meaning that the trader does not currently wish to trade and will not be held to buying or selling at any price.

Alternatively, we also use the following functions to denote the subfields of a shout tuple

$$\begin{aligned} \mathrm{price}(\rho) &= \rho_p \\ \mathrm{class}(\rho) &= \rho_c \\ \mathrm{agent}(\rho) &= \rho_a \\ \mathrm{time}(\rho) &= \rho_t \end{aligned}$$

3.1.3 Active traders

The finite set $K_t = \{k_{t1}, k_{t2}, \ldots, k_{tn}\}$ denotes the traders who are eligible to place shouts in the auction at time t. We pick the next trader whose turn it is to shout, τ_t, randomly from this set:

$$\tau_t = k_{t\delta_t}$$

where $\delta_t \in \mathbb{N}$ is a discrete random variable distributed according to a uniform distribution on the interval $[1, |K_t|]$, and we then remove this trader from the active set:

$$K_{t+1} = K_t - \tau_t$$

3.1.4 Events

Some of our state variables change in response to *events*. The possible types of event in our market are represented by the set:

$$\epsilon = \{eor, eod, sp, clr\}$$

These events denote “the end of a round”, “the end of a day”, “shout placed” and “market clearing” respectively, and are defined formally later. Events are time-stamped according to the time-slice at which they occurred. We denote this by subscripting events thus:

$$\epsilon_t = \{eor_t, eod_t, \dots\}$$

Thus, we have:

$$\begin{aligned}\epsilon_1 &= \{eor_1, eod_2, \dots\}\\ \epsilon_2 &= \{eor_2, eod_2, \dots\}\end{aligned}$$

The set E_t denotes the set of events that *occurred* at time t, as well as the set of events that were previously active in prior time slices. An event x_t *occurred* at time t if, and only if $x_t \in E_t$.

3.1.5 The end of round event

The end of round event, eor, is defined thus:

$$\begin{aligned}K_t = \{\} &\implies eor_{t+1} \in E_{t+1}\\ eor_t \in E_t &\implies K_{t+1} = A \wedge\ round_{t+1} = round_t + 1\end{aligned}$$

That is, the end of round event occurs once all traders have submitted offers, and when this event occurs we reset K to allow all traders to submit shouts in the next round.

3.1.6 Shout processing

The auctioneer maintains four sets of shouts. The sets $\hat{MS}_t$ and $\hat{MB}_t$ represent the set of matched asks and matched bids respectively.

We denote the i^{th} highest matched bid at time t by $\hat{mb}_{(t,i)}$, where

$$\text{price}(\hat{mb}_{(t,1)}) \geq \text{price}(\hat{mb}_{(t,2)}) \geq \text{price}(\hat{mb}_{(t,3)}) \geq \dots$$

Similarly, for matched asks we have:

$$\text{price}(\hat{ms}_{(t,1)}) \leq \text{price}(\hat{ms}_{(t,2)}) \leq \text{price}(\hat{ms}_{(t,3)}) \leq \dots$$

The match sets are maintained such that the following constraints hold:

$$\forall i\ \text{price}(\hat{mb}_{(t,i)}) \geq \text{price}(\hat{ms}_{(t,i)}) \quad (3)$$

$$|\hat{MS}_t| = |\hat{MB}_t| \quad (4)$$

The sets $\hat{MS'}_t$ and $\hat{MB'}_t$ contain all unmatched shouts at time t. Intuitively, the sets $\hat{MS}_t$ and $\hat{MB}_t$ can be thought of as the potential “winning” shouts at time t, and the sets $\hat{MS'}_t$ and $\hat{MB'}_t$ as the “runner-up” or “outbid” shouts at time t.

Let ρ denote the shout submitted to the auctioneer by τ_t — the trader who is currently shouting. These sets are updated as follows:

$$\begin{aligned}\rho_c = bid \wedge (\exists a \in \hat{MS'}_t : \rho_p \geq a_p) \implies \\ \hat{MS}_{t+1} = \hat{MS}_t \cup \{a\} \\ \wedge\ \hat{MS'}_{t+1} = \hat{MS'}_t - \{a\} \\ \wedge\ \hat{MB}_{t+1} = \hat{MB}_t \cup \{\rho\}\end{aligned} \tag{5}$$

$$\begin{aligned}\rho_c = bid \wedge (\nexists a \in \hat{MS'}_t : \rho_p \geq a_p) \implies \\ \hat{MB'}_{t+1} = \hat{MB'}_t \cup \{\rho\}\end{aligned} \tag{6}$$

$$\begin{aligned}\rho_c = ask \wedge (\exists b \in \hat{MB'}_t : b_p \geq \rho_p) \implies \\ \hat{MB}_{t+1} = \hat{MB}_t \cup \{b\} \\ \wedge\ \hat{MB'}_{t+1} = \hat{MB'}_t - \{b\} \\ \wedge\ \hat{MB}_{t+1} = \hat{MB}_t \cup \{\rho\}\end{aligned} \tag{7}$$

$$\begin{aligned}\rho_c = ask \wedge (\nexists b \in \hat{MB}'_t : b_p \geq \rho_p) \implies \\ \hat{MS'}_{t+1} = \hat{MB'}_t \cup \{\rho\}\end{aligned} \tag{8}$$

$$\rho_c \neq \emptyset \implies sp \in E_{t+1} \tag{9}$$

3.1.7 Quotes

The auctioneer provides the following information about the state of the auction:

$$\hat{eq}_a(t) = \min(\min(\hat{MS'}_t), \min(\hat{MB}_t)) \tag{10}$$
$$\hat{eq}_b(t) = \max(\max(\hat{MS}_t), \max(\hat{MB'}_t)) \tag{11}$$

The pair $(\hat{eq}_a(t), \hat{eq}_b(t))$ is called the *market quote*, and is public information to all traders participating in the market.

3.1.8 Trading days

A trading day consists of a number of rounds of trading. Different events may take place at the end of a day depending on the scenario we are modelling. For example, in many scenarios we will allocate new randomly drawn valuations for traders at the end of each trading day. These conditions will be introduced later. For now, we introduce the variable day_t which denotes the current trading day:

$$\begin{aligned} eod_t \in E_t &\implies day_{t+1} = day_t + 1 \\ \neg eod_t \in E_t &\implies day_{t+1} = day_t \end{aligned}$$

3.1.9 The clearing operation

The key role of the auctioneer is to compute a payment set C_t and a transaction set R_t as a function of the auction state $(\hat{MS}_t, \hat{MB}_t, \hat{MS'}_t, \hat{MB'}_t)$. Different variants of the double-auction mechanism compute C_t differently in order to bring about different design objectives. The specific variant we discuss in this paper is the *clearing-house* (CH) mechanism [10, p. 5] with uniform-pricing, in which the auctioneer batches up shouts from multiple traders before computing a clearing price which applies to all trades. These rules are formalised as follows.

A uniform pricing policy specifies that all traders with matched offers (that is, all the potentially efficient trades) should all trade with each other at the same price computed as a function of the market quote (as determined by $\hat{eq}_a$ and $\hat{eq}_b$). Thus, at any given time, all traders are transacting at the same global market price (which may change over time):

$$\begin{array}{rcl}
clr_t \in E_t \implies & & \\
\Gamma_{t+1} & = & \mathrm{pay}(C_t, \Gamma_t) \\
\wedge\ \Omega_{t+1} & = & \mathrm{trans}(R_t, \Omega_t) \\
\wedge\ M\hat{S}_{t+1} & = & \{\} \\
\wedge\ M\hat{B}_{t+t} & = & \{\} \\
\neg clr_t \in E_t \implies & & \\
C_t & = & \{\} \\
\wedge\ R_t & = & \{\}
\end{array}$$

where:

$$\forall_{i \leq |\hat{MB}|}\ c_i = (\mathrm{agent}(\hat{mb}_{(t,i)}), \mathrm{agent}(\hat{ms}_{(t,i)}), p_t)$$

and:

$$p_t = \hat{eq}_a(t)k + \hat{eq}_b(t)(1-k)$$

where $k \in [0,1]$ is a constant chosen by the market designer. In this paper we use a $k = \frac{1}{2}$ mechanism.

In a CH mechanism, the clearing operation is scheduled at the end of every round:

$$\begin{array}{rcl}
eor_t \in E_t & \implies & clr_{t+1} \in E_{t+1} \\
clr_t \in E_t & \implies & C_t = \{c_1, c_2, \dots\}
\end{array}$$

4 Search space

In the previous section we described in detail the game to which we apply our method for strategy acquisition. In this section we describe a space of strategies for this game.

Each agent a_i has an associated trading strategy, which specifies a mapping Z between its valuation v_i and the shout $\rho \in P$ that it will place at time t. For simplicity, we shall assume that: buyers always submit bids, sellers always submit asks, each agent only submits shouts for a single unit, and only the active traders K_t place shouts (see 3.1.3) . Thus:

$$\begin{aligned} Z(i,t) = (bid, a_i, \zeta(i,t), 1, t) &\iff a_i \in B \wedge a_i \in K_t \\ Z(i,t) = (ask, a_i, \zeta(i,t), 1, t) &\iff a_i \in S \wedge a_i \in K_t \\ Z(i,t) = (\emptyset, a_i, 0, 0, t) &\iff a_i \notin K_t \end{aligned}$$

where ζ is a function that sets the *price* of the shout according to the strategy being deployed.

4.1 The Truth-Telling Strategy

The truth-telling strategy (abbreviation TT) simply places shouts equal to the agent's valuation:

$$\zeta(i,t) = v_i \tag{12}$$

Although it is extremely simple, the truth-telling strategy is of fundamental importance, since in an *incentive-compatible* mechanism by definition this strategy is guaranteed to obtain the optimal payoff for agent a_i no matter what strategies are adopted by the other agents [13].

4.2 The Gjerstad-Dickhaut strategy

The Gjerstad-Dickhaut (abbreviation GD) strategy estimates the probability of a shout being accepted based on historical observations and then places its shout to maximise the agent's expected profit [11].

Agents using the GD strategy make use of a memory mechanism that records the shouts that gave rise to the last n transactions in the market, where $n = GD_N \in \mathbf{N}$ is the parameter that determines the size of the memory. The memory is divided into four sets:

$\hat{HS}_t \subset P$	The history of accepted asks up until time t
$\hat{HB}_t \subset P$	The history of accepted bids up until time t
$\hat{HS'}_t \subset P$	The history of unaccepted asks up until time t
$\hat{HB'}_t \subset P$	The history of unaccepted bids up until time t

The history is empty at the start of trading:

$$\hat{HS}_0 = \hat{HB}_0 = \hat{HS'}_0 = \hat{HB'}_0 = \{\} \tag{13}$$

As shouts are *placed* (Section 3.1.6) they are recorded in the history of *unaccepted* shouts:

$$\hat{HS'}_{t+1} = \hat{HS'}_t \cup \rho \iff \rho \in \hat{MS'}_t \tag{14}$$

$$\hat{HB'}_{t+1} = \hat{HB'}_t \cup \rho \iff \rho \in \hat{MB'}_t \tag{15}$$

As shouts are *matched* (Section 3.1.6) they are recorded in the history of *accepted* shouts:

$$\hat{HS}_{t+1} = \hat{HS}_t \cup \rho \iff \rho \in \hat{MS}_t \tag{16}$$

$$\hat{HB}_{t+1} = \hat{HB}_t \cup \rho \iff \rho \in \hat{MB}_t \tag{17}$$

Note that the history is unaffected by the clearing operation (Section 3.1.9), hence once a shout is recorded as accepted it remains so, unless it is removed due to memory-size restrictions as defined below.

Let

$$\vec{hs}_t = \{hs_{(1,t)}, hs_{(2,t)}, \dots, hs_{(GD_N,t)}\} \tag{18}$$

where $hs_{(1,t)} \in \mathbb{N}$ represents the total number of asks that were recorded before the 1^{st} most recent transaction, $hs_{(2,t)}$ is the total number of asks before the 2^{nd} most recent transaction *etc*.

Similarly let

$$\vec{hb}_t = \{hb_{(1,t)}, hb_{(2,t)}, \dots, hb_{(GD_N,t)}\} \tag{19}$$

where $hb_{(1,t)} \in \mathbb{N}$ represents the total number of bids that were recorded before the 1^{st} most recent transaction, $hb_{(2,t)}$ is the total number of bids before the 2^{nd} most recent transaction *et cetera*.

Let the scalar $h_t \in [0, GD_N)$ represent the current transaction number defined as follows

$$clr_t \in E_t \implies h_{t+1} = h_t + |C_t| \mod GD_N \tag{20}$$

$$\begin{aligned} &\exists \rho : \rho_t = t \wedge \rho_c = ask \implies \\ &hs_{(h_t+1,t+1)} = hs_{(h_t+1,t)} + 1 \end{aligned} \tag{21}$$

Agents using the GD strategy use the history data to form an estimate, $GD_{\hat{pa}(p)}$ of the probability of a shout with price p being accepted, based on:

- the number of asks accepted at prices greater than or equal to p;

$$GD_{TAG(p,t)} = |\{\rho : \rho \in \hat{HS}_t \wedge \rho_p \geq p\}| \tag{22}$$

- the total number of bids in the history at prices greater than or equal to p;

$$GD_{BG(p,t)} = |\{\rho : \rho \in (\hat{HB}_t \cup \hat{HB}'_t) \wedge \rho_p \geq p\}| \tag{23}$$

- the number of rejected asks in the history at prices less than or equal p;

$$GD_{RAL(p,t)} = |\{\rho : \rho \in \hat{HS}'_t \wedge \rho_p \leq p\}| \tag{24}$$

- the number of accepted bids at prices less than or equal to p;

$$GD_{TBL(p,t)} = |\{\rho : \rho \in \hat{HB}_t \wedge \rho_p \leq p\}| \tag{25}$$

- the total number of asks in the history at prices less than or equal to p;

$$GD_{AL(p,t)} = |\{\rho : \rho \in (\hat{HS}_t \cup \hat{HS}_t^{'}) \wedge \rho_p \leq p\}| \tag{26}$$

- and the number of rejected bids at prices greater than or equal to ρ_p

$$GD_{RBG(p,t)} = |\{\rho : \rho \in |\{\rho \in \hat{HB}_t' \wedge \rho_p \geq p\}| \tag{27}$$

Where we have recorded an ask at price p in the history (i.e., $\exists \rho : \rho \in (\hat{HS}_t \cup \hat{HS}_t^{'}) \wedge \rho_p = p$), the estimated probability of a new ask being accepted at the same price is given by the following equation:

$$GD_{\hat{pa}(p,t)} = \frac{GD_{TAG(p,t)} + GD_{BG(p,t)}}{GD_{TAG(p,t)} + GD_{BG(p,t)} + GD_{RAL(p,t)}} \tag{28}$$

Similarly, where we have recorded a bid at price p in the history, the estimated probability of a new bid being accepted is:

$$GD_{\hat{pa}(p,t)} = \frac{GD_{TBL(p,t)} + GD_{AL(p,t)}}{GD_{TBL(p,t)} + GD_{AL(p,t)} + GD_{RBG(p,t)}} \tag{29}$$

For prices not recorded in the history, the function

$$GD_{pa(p,t)} = \alpha_{(3,t)}p^3 + \alpha_{(2,t)}p^2 + \alpha_{(1,t)}p + \alpha_{(0,t)}$$

is obtained using cubic-spline interpolation over the pairs defined by the function $GD_{\hat{pa}(p,t)}$.

Now that we have an estimate of the probability of a shout being accepted at a particular price, we are in a position to estimate the expected surplus as a result of bidding at different prices. For buyer i:

$$GD_{E(p,i,t)} = (v_i - p_p)GD_{pa(p)} \tag{30}$$

and for seller i:

$$GD_{E(p,i,t)} = (p_p - v_i)GD_{pa(p)} \tag{31}$$

Finally, the GD strategy chooses prices in order to maximise expected surplus:

$$\zeta(i,t) = \arg\max_{p*} GD_{E(p*,i,t)} \tag{32}$$

4.3 Reinforcement-learning Strategies

Reinforcement-learning strategies rely only on the immediate feedback from interacting with the mechanism; the surplus that each agent was able to obtain in the most recent round of trading (thus they are general-purpose enough to be used in any auction-mechanism, even where we do not have access to market-data, for example, in repeated *sealed-bid* auctions).

These strategies choose their markup over their valuation price thus:

$$\zeta(i,t) = v_i + RL_{\lambda_i}(t)RL_{\mu_i} \iff a_i \in S \tag{33}$$

$$\zeta(i,t) = v_i - RL_{\lambda_i}(t)RL_{\mu_i} \iff a_i \in B \tag{34}$$

based on a *reward signal* $RL_{\rho_i}(t)$ which represents the most recent profits of agent a_i:

$$RL_{\rho_i}(t) = \Gamma_t(a_i) - \Gamma_{t-1}(a_i) \tag{35}$$

The function $RL_{\lambda_i} : \mathbb{N} \to \Theta_i$ represents the output of learning algorithm λ where $\Theta_i = [0, RL_{k_i}) \subset \mathbb{N}$ is the set of possible outputs from λ.

Parameter name	*Semantics*
$RL_{\lambda_i}(t)$	A function specifying the output from a reinforcement learning algorithm
RL_{μ_i}	A scaling factor used to map learning outputs onto actual prices
RL_{k_i}	The number of possible outputs from RL_{λ_i}

Table 1: Reinforcement-learning parameters

4.3.1 The Dumb-Random learning algorithm

The dumb-random learning algorithm (abbreviation DR) is a control algorithm that in fact performs no learning and chooses actions randomly:

$$RL_{\lambda_i} = \delta_{i_t} \tag{36}$$

where δ_{i_t} is a discrete random variable distributed uniformly in the range $[0, RL_{k_i})$. This algorithm can be used in control experiments by substituting it for one of the other algorithms below; if an observation is preserved under this substitution we can conclude that our observation is not likely to be due to learning behaviour.

4.3.2 The Roth-Erev learning algorithm

The Roth-Erev algorithm (abbreviation RE) is designed to mimic human game-playing behaviour in extensive form games [5]. Agents bid probabilistically according to:

$$RL_{\lambda_i}(t) = RE_i(t) = \delta_{i_t} \tag{37}$$

where $\delta_{i_t} \in \Theta_i$ is a discrete random variable distributed:

$$P(\delta_{i_t} = x) = RE_p(x, i, t) \tag{38}$$

The propensities are initialised based on the scaling parameter RE_{s_i}; $\forall a_i \in A$ and $\forall \theta \in \Theta_i$:

$$RE_q(\theta, a_i, t_0) = \frac{RE_{s_i}}{RL_{k_i}} \tag{39}$$

the RE_q are then updated based on the experience function RE_ϵ:

$$\begin{aligned} RE_q(\theta, a_i, t) = (1 - RE_{\rho_i}) RE_q(\theta, a_i, t-1) \\ + RE_\epsilon(\theta, a_i) \end{aligned} \tag{40}$$

where the experience function depends on the most recent reward signal RL_ρ and the last action chosen by the agent $RE_i(t-1)$:

$$\begin{aligned} RE_\epsilon(\theta, a_i, t) = \quad & RL_{\rho_i}(t-1)[1 - RE_{\eta_i}] \\ & \iff \theta = RE_i(t-1) \end{aligned} \tag{41}$$

$$\begin{aligned} RE_\epsilon(\theta, a_i, t) = \quad & RL_{\rho_i}(t-1)\tfrac{RE_{\eta_i}}{RL_{k_i}-1} \\ & \iff \theta \neq RE_i(t-1) \end{aligned} \tag{42}$$

and then normalized to produce a vector of probabilities; let Q_{i_t} denote the sum of all the propensities for agent i:

$$Q_{i_t} = \sum_{\theta \in \Theta_i} RE_q(\theta, a_i, t) \tag{43}$$

Then $\forall \theta \in \Theta_i$ and $\forall a_i \in A$:

$$RE_p(\theta, a_i, t) = \frac{RE_q(\theta, a_i, t)}{Q_{i_t}} \tag{44}$$

Parameter name	*Semantics*
RE_{k_i}	The number of possible outputs
RE_{ρ_i}	The recency parameter
RE_{η_i}	The experimentation parameter η
RE_{s_i}	The scaling parameter

Table 2: Parameters for the Roth-Erev learning algorithm

State variable	*Semantics*
$RE_i(t)$	The output of the learning algorithm at time t
$RE_p(\theta, a_i, t)$	The probability distribution over each possible action $\theta \in \Theta_i$
$RE_q(\theta, a_i, t)$	The *propensity* for each possible action $\theta \in \Theta_i$
$RE_\epsilon(\theta, a_i, t)$	The experience function

Table 3: State variables for the Roth-Erev learning algorithm

4.3.3 *Nicolaisen* et al.*'s modified Roth-Erev algorithm*

Nicolaisen, Petrov and Tesfatsion [18] (abbreviation NPT) used a modified version of the Roth-Erev algorithm for their trading strategy which they used to explore market power effects in a simulated electricity market:

$$RL_{\lambda_i}(t) = RE'_i(t) \tag{45}$$

where $RE'_i(t)$ is computed identically to $RE_i(t)$ but for a modification to the experience function:

$$\begin{aligned} RE_{\epsilon'}(\theta, a_i, t) = \quad & RL_{\rho_i}(t-1)[1 - RE_{\eta_i}] \\ & \iff \theta = RL_I(t-1) \end{aligned} \tag{46}$$

$$\begin{aligned} RE_{\epsilon'}(\theta, a_i, t) = \quad & RE_{q_i} \tfrac{RE_{\eta_i}}{RL_{k_i}-1} \\ & \iff \theta \neq RE_i(t-1) \end{aligned} \tag{47}$$

4.3.4 *The Stateless Q-Learning algorithm*

The Stateless Q-learning algorithm (abbreviation SQ) is a single-state version of a temporal-difference reinforcement-learning algorithm called Q-Learning [31]. The algorithm maintains a table $SQ_Q(\theta, a_i, t)$ which can be thought of as an estimate of the payoff to each possible action $\theta \in \Theta_i$. The estimates are updated using the rule:

$$\begin{aligned} & SQ_Q(\theta, a_i, t+1) = \\ & \quad SQ_Q(\theta, a_i, t) + \\ & \quad SQ_{\alpha_i}\left[RL_{\rho_i} + SQ_{\gamma_i} \max_{\theta'} SQ_Q(\theta', a_i, t) - SQ_Q(\theta, a_i, t)\right] \end{aligned} \tag{48}$$

where $SQ_{\gamma_i} \in \mathbb{R}$ is a discount factor and SQ_{α_i} is a parameter controlling the rate of convergence.

Actions are chosen to maximise estimated payoff using an ϵ-greedy rule:

$$\begin{aligned} RL_{\lambda_i}(t) = \delta_{it} \quad & \iff \quad \epsilon'_{it} \leq SQ_{\epsilon_i} \\ RL_{\lambda_i}(t) = \arg\max_{\theta*} SQ_Q(\theta*, a_i, t) \quad & \iff \quad \epsilon'_{it} > SQ_{\epsilon_i} \end{aligned}$$

where $\epsilon'_{it} \in \mathbb{R}$ is a random variable distributed uniformly on the interval $[0, 1]$ and $\delta_{it} \in \mathbb{N}$ is a discrete random variable distributed uniformly on the interval $[0, RL_{k_i} - 1]$.

Parameter name	*Semantics*
SQ_{ϵ_i}	The exploration parameter
SQ_{γ_i}	The discount factor
SQ_{α_i}	The learning rate

Table 4: Parameters for the stateless Q-Learning algorithm

5 Interaction between strategies

In the previous section we described a space of strategies for the double-auction. In this section we analyse the strategic-interaction between a representative subset of these strategies using the empirical game-theory methodology described in Section 2.4. As in [29], at the start of each game half the agents are randomly assigned to be buyers and the remainder as sellers. For each run of the game, valuations are drawn as in [29]:

$$\begin{array}{rcl} \forall_i\ v_i & \sim & U(a, a+b) \\ a & \sim & U(161, 260) \\ b & \sim & U(60, 100) \end{array}$$

but valuations remain fixed across periods in order to allow agents to attempt to learn to exploit any market-power advantage in the supply and demand curves defined by the limit prices for that game. Additionally, although we discard limit-prices which do not yield an equilibrium price, we do not ensure that a minimum quantity exists in competitive equilibrium as this introduces a floor effect which fails to expose the inferior efficiency of a CDA. The 64-bit version of the Mersenne Twister random number generator [15] was used to draw all random values used in the simulation and all floating point calculations were performed using IEEE 754 double-precision arithmetic [27]. Each entry in the heuristic payoff matrix was computed by averaging the payoff to each strategy across 10^4 simulations.

We use the representative strategies TT, RE, GD as described in table 5: the TT strategy was chosen since it is the simplest strategy that is able to achieve high efficiency outcomes in a homogenous population in the CH mechanism; the GD strategy was chosen as a representative of the class of highly-principled and highly-engineered strategies that analyse historical market data, and finally the RE strategy was chosen to represent naive human-like behaviour, and thus was configured with parameters that best-fit human game-playing [24]:

$$\begin{array}{rcl} \forall i\ RE_{k_i} & = & 50 \\ \forall i\ RE_{\rho_i} & = & 0.1 \\ \forall i\ RE_{\eta_i} & = & 0.2 \\ \forall i\ RE_{s_i} & = & 9 \\ \forall i\ RL_{\mu_i} & = & 1 \end{array}$$

Abbreviation	*Description*
TT	The truth-telling strategy, (section 4.1)
RE	The reinforcement-learning strategy (section 4.3), configured with Roth-Erev (section 4.3.2)
GD	The Gerstad-Dickhaut strategy (section 4.2)

Table 5: The initial heuristic strategies chosen for the analysis

In a conventional game-theoretic analysis, we solve the game by finding either a dominant strategy or the Nash equilibria: the sets of strategies that are best-responses to each other. However, because classical game-theory is a static analysis, it is not able to make any predictions about which equilibria are more likely to occur in practice. Such considerations are of vital importance in analysing real-world problems. For example, if we are interested in using game-theory to analyse economic outcomes, we should give more consideration to outcomes that are more likely than low probability outcomes; if there is a Nash equilibrium for our mechanism which yields very low allocative efficiency, we should not worry too much if this equilibria is extremely unlikely to occur in practice. On the other hand, we should give more weight to equilibria with high probability.

As in [29], we will use *evolutionary* game-theory [16] to model how agents might gradually adjust their strategies over time as they learn to improve their behavior in response to their payoffs. We use the replicator dynamics equation (Equation 1), to recap:

$$\dot{m}_j = [u(e_j, \vec{m}) - u(\vec{m}, \vec{m})]\, m_j$$

where $\vec{m}$ is a mixed-strategy vector, $u(\vec{m}, \vec{m})$ is the mean payoff when all players play $\vec{m}$, and $u(e_j, \vec{m})$ is the average payoff to pure strategy j when all players play $\vec{m}$, and $\dot{m}_j$ is the first derivative of m_j with respect to time. Strategies that gain above-average payoff become more likely to be played, and this equation models a simple *co-evolutionary* process of mimicry learning, in which agents switch to strategies that appear to be more successful. Since mixed strategies represent probability distributions, the components of $\vec{m}$ sum to one. The geometric corollary of this is that the vectors $\vec{m}$ lie in the *unit-simplex* $\triangle^n = \{\vec{x} \in \mathbb{R}^n : \sum_{i=1}^{n} x_i = 1\}$. In the case of $n = 3$ strategies the unit-simplex $\triangle^3$ is a *two*-dimensional plane triangle embedded in three-dimensional space which passes through the coordinates corresponding to pure strategy mixes: $(1, 0, 0)$, $(0, 1, 0)$, and $(0, 0, 1)$. We shall use a two dimensional projection of this triangle to visualise the replicator dynamics in the next section[4].

For any initial mixed-strategy we can find the eventual outcome from this co-evolutionary process by solving $\forall j,\ \dot{m}_j = 0$ to find the final mixed-strategy of the converged population. As discussed in Section 2.2, this has a significant advantage over non-game-theoretic co-evolutionary search, such as [12], in that we can *guarantee* [33, pp. 88–89]:

- all Nash equilibria of the (approximated) game are stationary points under the replicator dynamics; and
- all interior limit states are Nash equilibria; and
- all Lyapunov stable states [14] are Nash equilibria.

Thus the Nash equilibrium solutions are embedded in the stationary points of the direction field of the dynamics specified by Equation 1. Although not all stationary points are Nash equilibria, by overlaying a dynamic model of learning on the equilibria we can see which solutions are more likely to be discovered by *boundedly-rational* agents.

[4]See [33, pp. 3–7] for a more detailed exposition of the geometry of mixed-strategy spaces.

Those Nash equilibria that are stationary points at which a larger range of initial states will end up, are equilibria that are more likely to be reached (assuming an initial distribution of m_j that is uniform); in the terminology of dynamic systems they have a larger *basin of attraction*. The basin of attraction for a stationary point is proportion of mixed strategies in $\triangle$ which have flows terminating at that point[5]. The larger the basin, the larger the region of strategy-space which leads to the attractor, and hence the stronger the attractor, and the more *attainable* the corresponding equilibrium [1]. This intuitive definition of basin size is formalized as follows. Let the function

$$T : \triangle^n \times 2^{\triangle^n} \to \mathbb{N}$$

represent the trajectories that terminate at each coordinate in the n-dimensional unit-simplex $\triangle^n \subset \mathbb{R}^n$, so that we have:

$$\begin{aligned} & T(\vec{x}, M \subset \triangle^n) = \\ & \qquad |\{\vec{y} : \vec{y} \in M \wedge \vec{m}(0) = \vec{y} \wedge \exists t \; \vec{m}(t) = \vec{x} \wedge \dot{m}(t) = 0\}| \end{aligned} \tag{49}$$

where M is a set of starting points and $\vec{x}$ is a limit state. Let $\beta(\vec{x}, M)$ denote the *proportion* of the elements of M that terminate at $\vec{x}$:

$$\beta(\vec{x}, M) = \frac{T(\vec{x}, M)}{|M|} \tag{50}$$

If we choose a random sample $M \subset \triangle$ that is distributed uniformly over the simplex, the function β will provide us with an estimate of the probability of arriving at any given stationary point, assuming that all starting points in the simplex are equally likely; that is, it will provide an estimate of the true basin size of the limit state $\vec{x}$, denoted by $\beta(\vec{x})$, and:

$$\lim_{M \to \triangle} \beta(\vec{x}, M) = \beta(\vec{x})$$

Figure 1 shows the direction-field of the replicator-dynamics equation for our three heuristic strategies. In this and the subsequent direction-field diagrams, the points in the simplex represent alternative mixed and pure strategies, and the arrows indicate the direction of convergence when any such strategy is adopted. The three pure strategies (here, TT, RE and GD) are represented by the three vertexes of the simplex. A point on an external edge of the simplex represents a mixed strategy comprising two of the three pure strategies, and a point strictly inside the simplex represents a mixed strategy comprised of all three pure strategies. Thus, for example, the point on the left-most edge between the vertexes labeled TT and RE which is one-third the way from the vertex labeled TT represents a mixed strategy where strategy TT is chosen 66.7% of the time, strategy RE is chosen 33.3% of the time, and strategy GD not chosen at all; this position on the simplex is denoted (66.7, 33.3, 0). A vector (a line with an arrow)

[5]In many cases this will be the *volume* of the state space which terminates at the attractor, and this provides a useful intuition for thinking about attractor strength. However, in the general case this definition breaks down. For example, if we have chaotic dynamics then a strange attractor may capture many flows, but the volume of its basin will be zero.

shows the likely direction of strategic play from any given initial position. In other words, if the arrows converge on some point in the simplex, this strategy represented by that point is the end-point of repeated interactions as the game proceeds.

Looking at Figure 1, we can see there are two points where the direction vectors converge: these two points correspond to alternative equilibrium solutions of the evolutionary game. The first such point is the vertex at the bottom right, labeled GD. Since this point represents a pure strategy, the fact that this point is also a convergence point indicates that GD is a best-response strategy to itself, i.e., a pure-strategy equilibrium. We can also see that this point has a very large *basin of attraction*; for any randomly-sampled initial configuration of the population most of the flows end up in the bottom-right-hand-corner. The second point in the simplex where direction vectors converge is on the left-most edge between the points labeled TT and RE. This point corresponds to a second equilibrium, but is a mixed-strategy equilibrium, with co-ordinates of (0.88, 0.12, 0). This point represents an 88% mix of strategy TT and a 12% mix of RE. However, the basin of attraction for this equilibrium is much smaller than for the pure-strategy GD equilibrium; only 6% of random starts terminate at this mixed equilibrium vs. 94% for pure GD. Hence, according to this analysis, we would expect most of the population of traders to adopt the GD strategy. Note also that neither of the vertexes labeled TT or RE are the convergence points of direction flows; this indicates that neither strategy is the best response to itself.

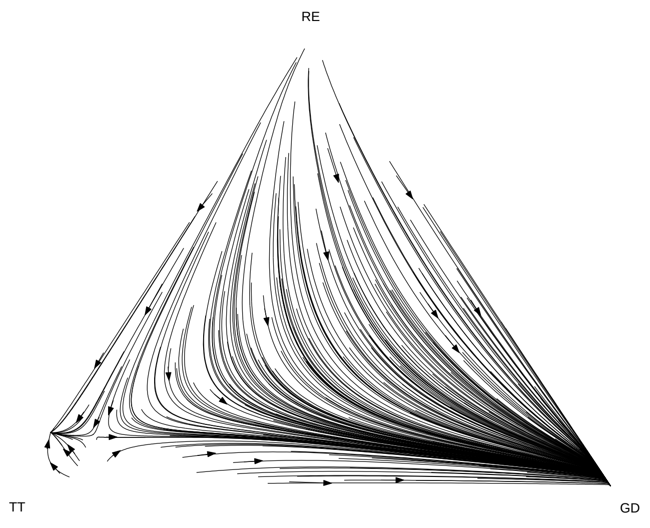

Figure 1: The original replicator dynamics direction field for a 12-agent clearing-house auction with the original unoptimized Roth-Erev strategy (labeled RE).

How much confidence can we give to this analysis given that the payoffs used to construct the direction-field plot were estimates based on simulation? One approach to answering this question is to conduct a sensitivity analysis; we perturb the mean payoffs for each strategy in the matrix by a small percentage to see if our equilibria analysis is robust to errors in the payoff estimates. Figure 2 shows the direction-field plot after

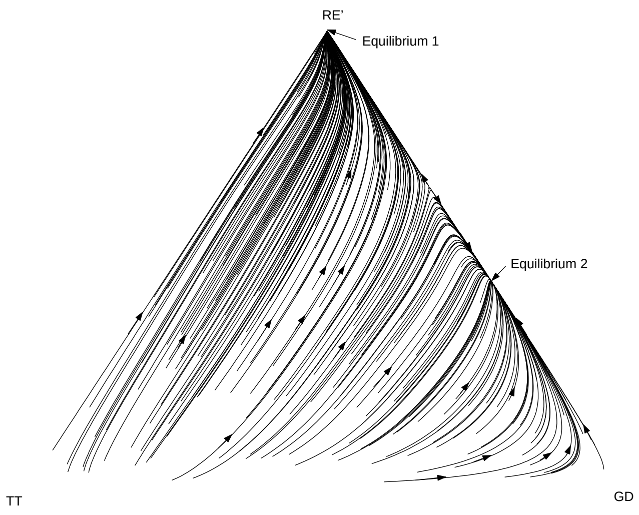

Figure 2: Replicator dynamics direction field for a 12-agent clearing-house auction perturbed with +5% payoffs to the Roth-Erev strategy (labeled RE')

performing a perturbation where 2.5% of the payoffs are removed from each of the TT and GD strategies and an additional +5% payoffs added to the RE strategy. This peturbation results in a qualitatively different set of equilibria. the two new equilibria are shown in Figure 2: one is a pure RE strategy, and the other a mix of GD and RE strategies. The RE strategy thus becomes a best-response to itself with a large basin of attraction (some 61%). We conclude from this peturbation analysis that the initial equilibrium analysis is sensitive to small changes or errors in payoff estimates, and so our initial prediction of widespread adoption of GD may not occur if the payoffs to RE have been under-estimated.

If we observe a mixture of all three strategies in actual play, however, the perturbation analysis also suggests that we could bring about widespread defection to RE if were able to tweak the strategy by improving its payoff slightly; *the perturbation analysis thus points to* RE *as a candidate for potential optimization*.

6 Strategy Acquisition

In the previous section we saw how heuristic-strategy approximation could be used to identify a potential candidate strategy for optimization. We also introduced an intriguing metric for ranking strategies on a single fully-ordered scale: viz, the size of the strategy's basin of attraction under the replicator dynamics. In this section we shall use this metric to perform a heuristic search of a space of strategies closely related to the RE strategy. In the following we shall define the space of strategies that are to be searched, and the details of the search algorithm.

The RE strategy discussed in the previous section belongs to a more general class of strategies: those based on reinforcement-learning. This class of strategies is described in detail in section 4.3. To recap, these strategies adjust their markup in response to the most recent profits obtained in the market using one of the following reinforcement learning algorithms: the Roth-Erev algorithm (RE), NPT's modifications to RE (NPT), the stateless Q-learning algorithm (SQ), and the control algorithm (DR). The parameters governing these algorithms are detailed in Tables 1 to 4.

Individuals in this search space were represented as a 50-bit string, where:

- bits 1-8 coded for parameter RL_μ in the range $(1, 10)$;
- bits 9-16 coded for the parameters SQ_ϵ or RE_η in the range $(0, 1)$;
- bits 17-24 coded for parameter RL_k in the range $(2, 258)$;
- bits 25-32 coded for parameters SQ_γ or RE_ρ in the range $(0, 1)$;
- bits 33-40 coded for parameter RE_s in the range $(1, 15000)$;
- bits 41-42 coded for the choice of learning algorithm amongst RE, NPT, SQ or DR; and
- bits 43-50 coded for parameter SQ_α in the range $(0, 1)$.

6.1 Search algorithm

A genetic-algorithm (GA) was used to search this space of strategies, where the fitness of each individual strategy in the search space was computed by estimating its basin size under the replicator dynamics under interaction with our existing three strategies: GD, TT and RE. As in Section 5, basin size was estimated using the function β defined in Equation 50, but since we recompute all entries in the heuristic-payoff matrix in support of each candidate strategy, we used lower sample sizes in order to facilitate evaluation of many strategies. The sample size for the number of games played for each entry in the heuristic payoff matrix was increased as a function of the generation number: $10 + int(100 \ln(g + 1))$ allowing the search-algorithm to quickly find high-fitness regions of the search-space in earlier generations and reducing noise and allowing more refinement of solutions in later generations. We used a constant number of replicator-dynamics trajectories $|M| = 50$ in order to estimate the basin size from the payoff matrix once it had been recomputed for our candidate strategy. Thus our fitness function is:

$$F(i, S, [H]) = \sum_{\vec{x} \in \epsilon_{[H]S}} \beta_{[H]}(\vec{x}, M) \cdot x_i \tag{51}$$

where: i is the index of the candidate heuristic strategy being evaluated from amongst the set of heuristic strategies S with heuristic payoffs $[H]$, $\beta_{[H]}$ denotes the basin size of an equilibrium in the game defined by payoffs $[H]$ as specified by Equation 50 (p. 21), and $\epsilon_{[H]S}$ is the set of heuristic equilibria:

$$\epsilon_{[H]S} = \{\vec{x} \in \triangle^{|S|} : \beta_{[H]}(\vec{x}, M) > 2 \times 10^{-2}\}$$

Since we are comparing with our three existing strategies, in this experiment we have:

$$S = \{\mathrm{s*}, \mathrm{TT}, \mathrm{GD}, \mathrm{RE}\}$$

where s* is our candidate strategy (i.e., $i = 1$). Thus the fitness function estimates the expected frequency with which our candidate strategy will be played in equilibrium outcomes. The entire search process is summarised in pseudo-code in Algorithm 1; we call this the `FiSH` algorithm, since we will use it to "fish" for a new heuristic strategy.

Algorithm 1: FiSH

```
input  : A set of heuristic strategies $S = \{s_1, s_2, \ldots s_n\}$
output: A new heuristic strategy OS
$[H] \leftarrow \mathrm{GetHeuristicPayoffMatrix}(S)$;
$\hat{F} \leftarrow 0$;
for $i \leftarrow 1$ to $n$ do
    $[H]' \leftarrow$ perturb payoffs in $[H]$ in favour of $s_i$;
    if $F(i, S, [H]') > \hat{F}$ then
        $\hat{F} \leftarrow F(i, S, [H]')$;
        $\hat{\mathrm{OS}} \leftarrow s_i$;
    end
end
$\Pi \leftarrow$ create a search space based on generalisations of $\hat{\mathrm{OS}}$;
$\mathrm{OS} \leftarrow \arg\max_{\mathrm{s*} \in \Pi} F(1, \mathrm{s*} \cup S, \mathrm{GetHeuristicPayoffMatrix}(\mathrm{s*} \cup S))$;
```

A GA was chosen to search the space Π of potential variations on RE, principally because of its ability to cope with the additional noise that the lower sample size introduced into the objective function. The GA was configured with a population size of 100, with single-point cross-over, a cross-over rate of 1, a mutation-rate of 10^{-4} and fitness-proportionate selection. The GA was run for 32 generations, which took approximately 1800 CPU hours on a dual-processor Xeon 3.6Ghz workstation.

7 Results

Figure 3 shows the mean fitness of the GA population for each generation. As can be seen, the variance in fitness values in later generations is still large. However, inspection of a random sample of strategies from each generation revealed a partial convergence of phenotype, but with significant fluctuations in fitness values due to small sample sizes (see above). Most notably, the fittest individual at generation 32 had also appeared intermittently as the fittest individual five times in the previous 10 generations, and thus this was taken as the output from the search.

The optimised strategy that evolved used the stateless Q-learning algorithm (SQ) with the following parameters:

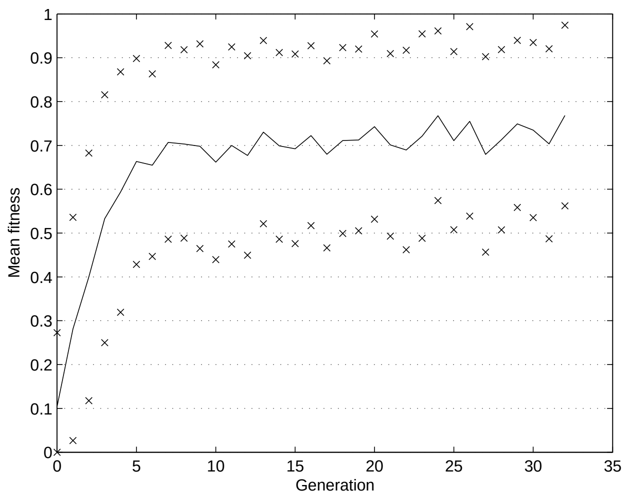

Figure 3: Mean fitness of the GA population with one standard deviation

$$
\begin{aligned}
RL_\mu &= 1.210937 \\
RL_k &= 6 \\
SQ_\epsilon &= 0.18359375 \\
SQ_\gamma &= 0.4140625 \\
SQ_\alpha &= 0.1875
\end{aligned}
$$

The notable feature of this strategy is the small number of possible markups RL_k, and the narrow range of the markups $[0, (RL_k - 1)RL_\mu]$ as compared with the distribution of valuation distribution widths. This feature was shared by all of the top five strategies in the last ten generations, and is another factor that indicated convergence of the search.

We proceeded to analyze our specimen strategy under a full heuristic-strategy analysis using 10^4 samples of the game for each of the 455 entries in the payoff matrix. Using *JASA*, the *Java Auction Simulator API*[6] developed by the first author [21], this analysis was completed in under twenty-four hours using a dual-processor 3.6Ghz Xeon workstation.

Figure 4 shows twenty trajectories of the replicator-dynamics plotted as a time-series graph for each strategy, and shows the interaction between the new, optimised strategy, OS, together with the existing strategies: GD, TT and RE.

[6]`http://freshmeat.net/projects/jasa`

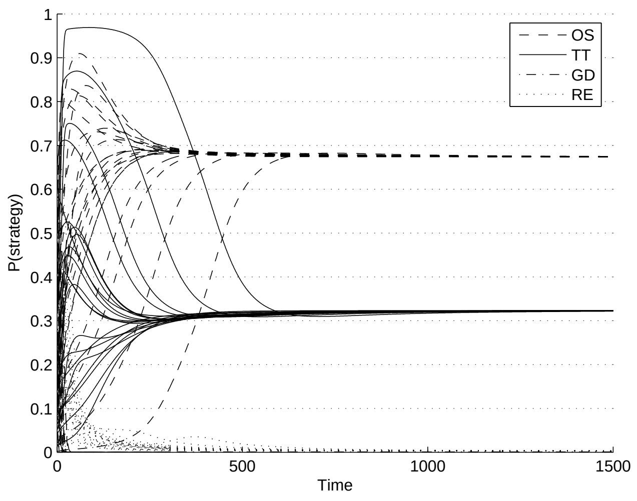

Figure 4: Replicator dynamics time series plot for a 12-agent clearing-house auction showing interaction between optimised strategy (OS) versus GD, TT and the original Roth-Erev strategy (RE)

Taking $M \subset \triangle^4 : |M| = 10^3$ randomly sampled initial mixed-strategies, we calculate that there are two attractors:

$$\begin{aligned} \vec{A} &= (0, 0, 1, 0) \\ \vec{B} &= (0.67, 0.32, 0, 0) \end{aligned}$$

over $(\mathrm{OS}, \mathrm{TT}, \mathrm{GD}, \mathrm{RE})$. Attractor A captures only

$$\beta(\vec{A}, M) = 0.03$$

that is, three percent of trajectories, whereas attractor B captures virtually the entire four-dimensional simplex:

$$\beta(\vec{B}, M) = 0.97$$

Although this basin is very large, our optimized strategy shares this equilibrium with the truth-telling strategy (TT), giving us a final total market share

$$F = 0.67 \times 0.97 = 0.65$$

This compares favourably with a market-share of 32% for truth-telling and 3% for GD. The original RE strategy is dominated by our optimised strategy. Figures 5 and 6 show the direction field for two of the 3-strategy combinations involving our optimised strategy: $(\mathrm{OS}, \mathrm{TT}, \mathrm{GD})$ and $(\mathrm{OS}, \mathrm{GD}, \mathrm{RE})$ respectively.

8 Discussion

It is somewhat remarkable that our fairly simplistic optimised strategy is able to gain defectors from a highly sophisticated strategy like GD, whilst at the same time truth-telling is able to retain a share of followers in a population predominated by OSers (TT appears to be *parasitic* on OS). What accounts for the ability of small OS mixes to invade high-probability mixes of a sophisticated adaptive strategy (GD), whilst remaining vulnerable to invasion by a low-probability mix of a non-adaptive strategy TT? A possible explanation is as follows.

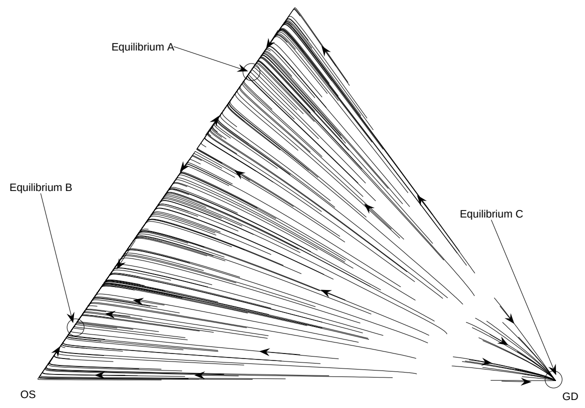

Figure 5: Replicator dynamics direction field for a 12-agent clearing-house auction showing interaction between optimised strategy (OS) versus TT and GD

As discussed earlier, we use the same method of assigning valuations as in [29]; that is, for each run of the game, the lower-bound, b, of the valuation distribution is selected uniformly at random from the range $[61, 160]$ and the upper-bound b' is similarly drawn from $[b + 60, b + 209]$. For that run of the game, each agent's valuation is then drawn uniformly from $[b, b']$. However, it is possible that this results in a statistical correlation between the meta-bounds and the average slope of truthful supply and demand schedules— that is, given these distribution parameters there is insufficient variance in the difference between valuations of traders who are neighbors on the supply or demand curve. Since we are using a uniform-price $k = 0.5$ clearing rule, the mechanism is vulnerable to price-manipulation from the least efficient trades; the buyer with the lowest matched bid, and the seller with the highest matched ask can potentially manipulate the final clearing price - *provided that they do not overstate their value claim to the extent that it impinges on the 2nd-lowest matched bid, or the 2nd-highest matched ask*. For example, in the case of buyer $a_i \in B$ who finds themselves with the lowest matchable valuation, and if we assume that the other agents are truth-tellers then our competitors' bids will be given by a subset of $MB = \{mb_1, mb_2, \ldots, mb_n\}$. The 2nd-lowest matched bid will be mb_{n-1} and our valuation will be given mb_n. Let:

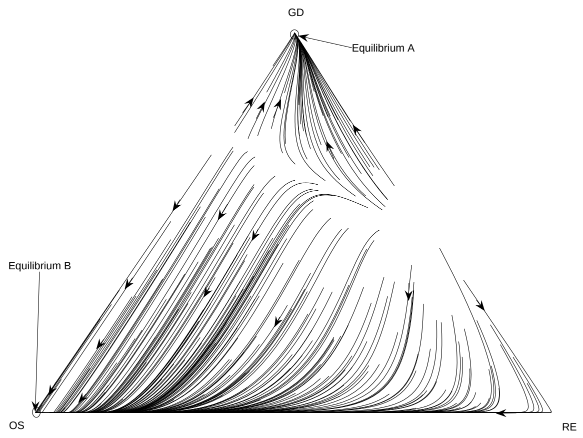

Figure 6: Replicator dynamics direction field for a 12-agent clearing-house auction showing interaction between optimised strategy (OS) versus GD and the original Roth-Erev strategy (RE)

$$\Delta mb = mb_{n-1} - mb_n$$

This is a random variable. However if we know the distribution of Δmb, we can calculate the probability of our bid being accepted as a function of its price: $P_{accept}(\hat{v_i})$. Since our profit will be $v_i - \hat{v_i}$, given knowledge of the distribution of Δmb it would be straightforward to choose a bid price $\hat{v_i}$ that maximises our expected profit:

$$\arg\max_{\hat{v_i}} E(U_i(\hat{v_i})) = (v_i - \hat{v_i}) P_{accept}(\hat{v_i})$$

Given sufficient variance in the distribution of Δmb this feature of the market is not easily exploited. However, in a market with a small number of traders and a narrow distribution for Δmb there is an opportunity to trade at small margin above truth if you find yourself with a valuation close to the equilibrium price $p*$. This is precisely the behaviour of the strategies that we observe to be predominant in the later generations of our GA: they all use a small number of possible markups, each of them small in comparison to the possible valuation bounds. The reinforcement-learning component of the strategy is then able to fine-tune the markup depending on where the trader finds themselves on the supply or demand curve after valuations are drawn. If the trader's valuation is far away from the equilibrium-price, the trader can adjust its margin close to zero, whilst if the trader's valuation is near to the equilibrium-price, the trader can find a small margin that does not impinge on its nearest-neighbour. This hypothesis is also consistent with parasitic truth-telling; it is easy to see that truth-telling is a best-response for a 2nd-lowest matched bidder to a lowest matched bidder playing OS.

In future work we will examine this hypothesis in more detail and conduct a statistical analysis in which we determine the distribution of Δmb for different parameters of the valuation distribution range, and attempt to correlate it with the parameters of the evolved strategy. Meanwhile, we have demonstrated that the search technique presented here is capable of finding a new strategy that not only has a large attractor, but also has interesting properties worthy of further analysis.

8.1 An iterative approach

We started out by asking whether our original equilibrium analysis of TT, GD and RE was sensitive to small perturbations in payoff estimates. By doing so, we identified that hypothetical variations on the RE strategy might be able to easily invade our existing equilibria. We then identified a new entrant OS that was able to penetrate the original mix of strategies and displace the ancestral incumbent RE, forming two new equilibria comprising mixes of OS, TT and GD. Thus by performing this analysis we have *refined* our original equilibrium analysis, since our original equilibria did not take into account the existence of OS. This process can be generalised to an arbitrary set of initial heuristic-strategies, as shown in Algorithm 1, the `FiSH` Algorithm.

We have validated `FiSH` empirically by applying it to a highly complex game, the double-auction, and demonstrated that it is capable[7] of finding a new strategy with interesting properties, as demonstrated in the previous section. However, one might ask whether our new strategy OS, or more accurately our new set of equilibria over $\mathrm{OS} \cup S$, is not susceptible to the same process of systematically searching for an invader? Of course, the answer is that this is indeed a possibility. We could straightforwardly test for this by applying exactly the same analysis to our new set of equilibria; that is, we could perform another sensitivity analysis to see whether our new equilibria are stable under payoff perturbation. If they were, then we might conclude that our equilibria are comparatively stable for the time being. If they are not stable, however, we could then perform another systematic search for variations in the current strategies which are good candidates for potential invaders of the status quo; that is, new strategies which form equilibria with estimated large basin size in interaction with the incumbents. By performing this process repeatedly we will eventually end up with a refined set equilibrium strategies. The pseudo-code for this process is shown in Algorithm 2, called the `FiSH+` Algorithm.

8.2 Applications

Many algorithms for strategy-acquisition focus on searching for strategies that are generally *robust* when played against existing strategies. However, as discussed in Section 2 it is extremely difficult to formulate objective metrics for ranking the robustness of strategies in the non-zero-sum n-player games which typify interactions in marketplaces and multi-agent systems. In contrast, our method for strategy acquisition focusses on searching for strategies that *are likely to be adopted* by the participants. This has several applications in both economics and computer science, which we discuss below.

[7]for at least one set of initial strategies $S = \{\mathrm{TT}, \mathrm{GD}, \mathrm{RE}\}$

Firstly, the level of adoption of a particular strategy may be a real-world design consideration in and of itself. For example, the inventor of a trading strategy such as ZIP [3] may have intellectual property rights that generate revenue in proportion to its level of adoption. In a wider context, many other software artifacts exist in a competitive ecology, and as Papadimitriou notes:

> *"If an artifact (a new congestion control protocol, a new caching scheme, a new routing algorithm, etc.) is demonstrated to have superior performance, this does not necessarily mean that it will be successful. For the artifact to be 'fit', there must exist a* path *leading from the present situation to its prevalence. This path must be paved with incentives that will motivate all kinds of diverse agents to adopt it, implement it, use it, interface with it or just tolerate it. In the absence of such a path, the most clever, fast and reliable piece of software may stay just that."* [19]

Secondly, the primary economic application of our method is to the *mechanism design* problem [13, 28, 23]. In a mechanism design problem one attempts to define market "mechanisms", that is, the rules of the market (Section 3.1), in such a way that design objectives such as maximising the market efficiency EA are achieved when agents follow their utility-maximising strategies. The revelation principle [13, p. 82] states that we can restrict this search problem to mechanisms in which agents directly reveal their valuations to the auctioneer; it then suffices to demonstrate that the TT strategy (Section 4.1) is a dominant strategy under our candidate mechanism (this property is called *incentive-compatibility*), and that efficiency, or other design objectives, are maximised when all agents adopt TT. However, real-world considerations mean that it is rarely possible to design incentive-compatible mechanisms in which a simple strategy such as TT is unequivocally dominant (and hence likely to be adopted), especially in the case of double-sided mechanisms, or when we have legacy constraints on design [20]. In such scenarios it may more practical to demonstrate that design-objectives such as high efficiency are satisfied when agents use an existing non-truthful strategy such as ZIP [3] or GD, provided that this strategy is *likely to to be adopted*. However, in many cases it will be difficult to demonstrate that a single existing strategy has a high probability of adoption. The `FiSH` algorithm can be used in precisely such a situation in order to search for highly-adoptable strategies [21].

Finally, there is a sense in which our algorithm may be useful for searching for robust strategies in non-zero-sum n-player games. In 2-player zero-sum games the Nash solution is guaranteed to yield the security level of the game, and is thus demonstrably robust. As discussed in Section 2 this result does not generalise to n-player non-zero-sum games. In such games, the best we can do is play a best-response to the strategies adopted by other agents; however, in the general case (i.e., with multiple equilibria) there is no unequivocal method that will tell us which strategies will be selected by our opponents. The `FiSH` algorithm escapes from this logic by searching for hitherto unconsidered strategies that are likely to be adopted by agents who *learn*. Thus if we modify Equation 51 to incorporate payoff maximisation in addition to basin size:

$$F'(i, S, [H]) = \sum_{\vec{x} \in \epsilon_{[H]S}} u(e_i, \vec{x}) \cdot \beta_{[H]}(\vec{x}, M) \cdot x_i \qquad (52)$$

we can then use the algorithm to find strategies that are simultaneously payoff-maximising and are also likely to be adopted by one's opponents (provided that they choose from the available strategies using a learning-process similar to that modelled by the replicator dynamics). In future work we will explore this application of our algorithm to more general games.

```
Algorithm 2: FiSH+
input  : A set of heuristic strategies $S = \{s_1, s_2, \ldots s_n\}$ for some
         mechanism $\mu$
output: A refined set of heuristic-strategies
$[H] \leftarrow \mathrm{GetHeuristicPayoffMatrix}(S, \mu)$;
repeat
    $\hat{F} \leftarrow \max_{i=1 \ldots n} F(i, S, [H])$;
    for $i \leftarrow 1$ to $n$ do
        $[H]' \leftarrow$ perturb payoffs in $[H]$ in favour of $s_i$;
        if $F(i, S, [H]') > \hat{F}$ then
            $\hat{F} \leftarrow F(i, S, [H]')$;
            $\mathrm{i}^* \leftarrow i$;
            $\hat{\mathrm{OS}} \leftarrow s_i$;
        end
    end
    if $\hat{F} < F(\mathrm{i}^*, S, [H])$ then return $S$;
    $\Pi \leftarrow$ create a search space based on generalisations of $\hat{\mathrm{OS}}$;
    $\mathrm{OS} \leftarrow$
    $\arg\max_{\mathrm{s}^* \in \Pi} F(1, \mathrm{s}^* \cup S, \mathrm{GetHeuristicPayoffMatrix}(\mathrm{s}^* \cup S, \mu))$;
    $S \leftarrow \mathrm{OS} \cup S$;
    $[H] \leftarrow \mathrm{GetHeuristicPayoffMatrix}(S, \mu)$;
    $S \leftarrow$ eliminate dominated strategies from $S$ based on $[H]$;
until forever ;
```

9 Conclusion

In this paper, we have introduced a novel method for acquisition of strategies in non-zero-sum n-player games, and have empirically validated this approach by applying it to a well-known benchmark problem, the double-auction market. Many existing approaches to strategy acquisition focus on attempting to find strategies that are robust in the sense that they are good all-round performers against any other strategy. We have argued that in many economic and multi-agent scenarios the robustness criterion is inappropriate and impossible to assess, due to the large number of possible strategies and the non-transitive relationships between these strategies. Instead, our method focusses on searching for strategies that are *likely to be adopted* by agents participating in the interaction, and then developing effective responses to these strategies.

The key strength of our proposed method for strategy acquisition is its ability to be applied in realistically complex games, such as the double-auction. However, just as the domain to which we have applied it suffers from a lack of analytic tractability, one potential weakness of the method is the lack of an analytical proof demonstrating its efficacy in the general case. However, this is mitigated by the fact that the single-iteration algorithm called `FiSH` combines two fields in a very simple way, each with a growing analytical literature, namely, empirical game-theory and optimisation. Additionally, we have demonstrated that this algorithm works effectively in at least one highly complex setting, thereby presenting an existence proof that the algorithm can be effective in a realistically-complex domain. For the empirical study in this paper we have used a general purpose optimisation method, i.e., a genetic algorithm. In future work we will attempt to find a specialised optimisation algorithm for the purposes of maximising attractor size by interleaving the optimisation and heuristic-strategy analysis steps in a similar manner to that proposed by Walsh *et al.* [30].

We have not attempted to validate the proposed iterative version of the algorithm, the `FiSH+` Algorithm, in this paper. Again, this algorithm is a fairly simple elaboration on the non-iterative version, so the lack of analytical validation should not detract from its potential. However, the fact that the approach is highly computationally intensive for a single iteration warrants an analysis of how the algorithm might converge prior to investing in a full empirical case study.

Acknowledgments

We are grateful for financial support received from the UK EPSRC through the *Market-Based Control of Complex Computational Systems* Project (GR/T10657/01) and the National Science Foundation under grant NSF IIS-0329037 *Tools and Techniques for Automated Mechanism Design*.

References

[1] S. Bullock. *Evolutionary Simulation Models: On Their Character, and Application to Problems Concerning the Evolution of Natural Signalling Systems*. PhD thesis, University of Sussex, 1997.

[2] J. Cartlidge and S. Bullock. Combating coevolutionary disengagement by reducing parasite virulence. *Evolutionary Computation*, 12(2):103–222, 2004.

[3] D. Cliff and J. Bruten. Minimal-intelligence agents for bargaining behaviors in market-based environments. Technical Report HPL-07-91, HP Labs, Bristol, August 1997.

[4] C. Daskalakis, P. W. Goldberg, and C. H. Papadimitriou. The complexity of computing a Nash equilibrium. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, 2006.

[5] I. Erev and A. E. Roth. Predicting how people play games: Reinforcement learning in experimental games with unique, mixed strategy equilibria. *American Economic Review*, 88(4):848–881, 1998.

[6] S. G. Ficici. *Solution Concepts in Coevolutionary Algorithms*. PhD thesis, Brandeis University, May 2004.

[7] S. G. Ficici and J. B. Pollack. Challenges in coevolutionary learning: Arms-race dynamics, open-endedness, and mediocre stable states. In *Proceedings of the Sixth International Conference on Artificial Life*, Cambridge, MA, 1998. MIT Press.

[8] S. G. Ficici and J. B. Pollack. A game-theoretic approach to the simple coevolutionary algorithm. In M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. J. Merelo, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature — PPSN VI*, pages 16–20, Berlin, Germany, 2000. Springer Verlag.

[9] S. G. Ficici and J. B. Pollack. A game-theoretic memory mechanism for coevolution. In E. Cant-Paz, J. A. Foster, K. Deb, D. Lawrence, R. Roy, U.-M. O'Reilly, H.-G. Beyer, R. Standish, G. Kendall, S. Wilson, M. Harman, J. Wegener, D. Dasgupta, M. A. Potter, A. C. Schultz, N. Jonoska, K. A. Dowslandand J. F., and Miller, editors, *Genetic and Evolutionary Computation — GECCO 2003*, pages 286–297, Berlin, Germany, 2003. Springer-Verlag.

[10] D. Friedman. The double auction institution: A survey. In D. Friedman and J. Rust, editors, *The Double Auction Market: Institutions, Theories and Evidence*, Santa Fe Institute Studies in the Sciences of Complexity, chapter 1, pages 3–25. Perseus Publishing, Cambridge, MA, 1993.

[11] S. Gjerstad and J. Dickhaut. Price formation in double auctions. *Games and Economic Behaviour*, 22:1–19, 1998.

[12] W. D. Hillis. Co-evolving parasites improve simulated evolution as an optimization procedure. In C. G. Langton, C. Taylor, J. D. Farmer, and S. Rasmussen, editors, *Artificial Life II*, Santa Fe Institute Studies in the Sciences of Complexity, pages 313–324. Perseus Publishing, Cambridge, MA, 1992.

[13] V. Krishna. *Auction Theory*. Academic Press, San Diego, CA, 2002.

[14] A. M. Lyapunov. *Stability of motion*. Academic Press, New York and London, 1966.

[15] M. Matsumoto and T. Nishimura. Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation*, 8(1):3–30, 1998.

[16] J. Maynard-Smith. *Evolution and the Theory of Games*. Cambridge University Press, Cambridge, UK, 1982.

[17] J. Nash. Equilibrium points in n-person games. In *Proceedings of the National Academy of Sciences*, volume 36, pages 48–49, 1950.

[18] J. Nicolaisen, V. Petrov, and L. Tesfatsion. Market power and efficiency in a computational electricity market with discriminatory double-auction pricing. *IEEE Transactions on Evolutionary Computation*, 5(5):504–523, October 2001.

[19] C. H. Papadimitriou. Algorithms, games, and the internet. In *Proceedings of the 33rd Symposium on Theory of Computing*, pages 749–753. ACM Press, 2001.

[20] S. Phelps, S. Parsons, and P. McBurney. An evolutionary game-theoretic comparison of two double-auction market designs. In P. Faratin and J. A. Rodriguez-Aguílar, editors, *Agent-Mediated Electronic Commerce VI*, pages 101–114. Springer Verlag, 2006.

[21] Steve Phelps. *Evolutionary Mechanism Design*. PhD thesis, University of Liverpool, 2008.

[22] D. M. Reeves, J. K. MacKie-Mason, M. P. Wellman, and A. Osepayshvili. Exploring bidding strategies for market-based scheduling. *Decision Support Systems*, 39:67–85, 2005.

[23] J. S. Rosenschein and G. Zlotkin. *Rules of Encounter: Designing Conventions for Automated Negotiation among Computers*. MIT press, 1994.

[24] A. E. Roth and I. Erev. Learning in extensive form games: experimental data and simple dynamic models in the intermediate term. *Games and Economic Behavior*, 8:164–212, 1995.

[25] J. Rust, J. H. Miller, and R. Palmer. Behaviour of trading automata in a computerized double auction market. In D. Friedman and J. Rust, editors, *The Double Auction Market: Institutions, Theories and Evidence*, Santa Fe Institute Studies in the Sciences of Complexity, chapter 6, pages 155–199. Perseus Publishing, Cambridge, MA, 1993.

[26] W. D. Smith. Rating systems for gameplayers, and learning. Technical Report 93-104-3-0058-5, NEC, 4 Independence Way, Princeton, NJ, 1994.

[27] D. Stevenson. IEEE task P754: A proposed standard for binary floating point arithmetic. *Computer*, 14(3):51–62, March 1981.

[28] H. R. Varian. Economic mechanism design for computerized agents. In *Proceedings of the First USENIX Workshop on Electronic Commerce*, pages 13–21, 1995.

[29] W. E. Walsh, R. Das, G. Tesauro, and J. O. Kephart. Analyzing complex strategic interactions in multi-agent games. In *Workshop on Game Theoretic and Decision Theoretic Agents*, 2002.

[30] W. E. Walsh, D. Parkes, and R. Das. Choosing samples to compute heuristic-strategy Nash equilibrium. In P. Faratin, D. C. Parkes, J. A. Rogdríguez-Aguilar, and W. E. Walsh, editors, *Agent-Mediated Electronic Commerce V: Designing Mechanisms and Systems*, pages 109–123, Berlin, Germany, 2003. Springer-Verlag.

[31] J. C. H. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8:279–292, 1992.

[32] R. A. Watson and J. B. Pollack. Coevolutionary dynamics in a minimal substrate. In L. Spector, E. D. Goodman, A. Wu, W. B. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. H. Garzon, and E. Burke, editors, *GECCO-2001: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 702–709, San Francisco, California, 2001. Morgan Kaufmann.

[33] J. W. Weibull. *Evolutionary Game Theory*. MIT Press, Cambridge, MA, 1997.

[34] M. P. Wellman. The economic approach to artificial intelligence. *ACM Computing Surveys*, 27(3), 1995.

[35] M. Wooldridge. *An Introduction to MultiAgent Systems*. John Wiley & Sons, 2002.

[36] P. R. Wurman, W. E. Walsh, and M. P. Wellman. Flexible double auctions for electronic commerce: theory and implementation. *International Journal of Decision Support Systems*, 24:17–27, 1998.

[37] W. Zhan and D. Friedman. Markups in double auction markets. *Journal of Economic Dynamics and Control*, 31(9):2984–3005, 2007.