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

Steve Phelps, Peter McBurney, Simon Parsons

***Abstract*—We introduce a 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, formalised as the size of a strategy's basins of attraction under the replicator dynamics.**

I. Introduction

We introduce a heuristic method for searching for strategies in multi-agent interactions such as auction marketplaces. Many existing heuristic methods for strategy acquisition, such as co-evolutionary search, attempt to search for strategies which yield high payoff irrespective of opponents' behaviour [1]. In contrast, our method searches for strategies that are likely to be adopted by a population of agents using social learning [2, p. 67]. We shall argue that the latter criteria is particularly useful in the context of a *mechanism design* [3] problem. 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 equilibrium strategies.

The traditional approach to mechanism design involves evaluating outcomes under conditions of Nash equilibria [4]. There are two main difficulties with this approach. Firstly, the traditional game-theoretic approach assumes that the space of strategies for each agent is common knowledge. However, in many realistic multi-agent interactions, the space of possible policies for each agent is not known apriori, making computation of the Nash equilibria intractable in the general case. Secondly, for arbitrary mechanisms 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.

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 [5] motivated much contemporary research in this area by highlighting the difficulty of agents' decision problems in non-idealized variants of this type of marketplace.

The outline of this paper is as follows. In section II we describe the double-auction game. In section III we detail the methodology that we use to analyse this game heuristically. In section IV we formalise a space of possible strategies for the double-auction. In section V we formalise the objective-function that we use to search for new strategies. In section VI we describe the search-space of strategies. In section VII we describe our search algorithm. In section VIII we report the results of an empirical validation of our algorithm. In section IX we discuss potential applications of our algorithm and we conclude in section X.

II. 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.

Our model of the double-auction is adapted from [6], [7], [8], [9], 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 mutually-excusive sets: *buyers*, represented by the set $B \subset A$; and *sellers*, represented by the set $S \subset A$. We assume that agents are risk-neutral (utility increases linearly with increased wealth). Buyers purchase resource for consumption, and sellers produce sellers for sale. Each agent a_i has a

S. Phelps is with the Centre for Computational Finance and Economic Agents, University of Essex, United Kingdom, e-mail: sphelps@essex.ac.uk.

P. McBurney is with the Department of Computer Science, University of Liverpool, United Kingdom, e-mail: mcburney@liverpool.ac.uk.

S. Parsons is with Department of Computer and Information Science, Brooklyn College, City University of New York, USA, e-mail: parsons@sci.brooklyn.cuny.edu.

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 $p - v_i$.

Trade is conducted through a bidding process in which agents submit *limit-orders* for a specified price, quantity, and direction (buy or sell). Limit-orders are analogous to bids in a single-sided auction; they specify the maximum (minimum) price at which an agent is willing to buy (sell). The double-auction uses a limit-order book to match orders from opposing directions, producing a set of *transactions* which determine the price, quantity and counter-parties of any given trade in the market. The process of producing transactions from orders is called *clearing*. There are many variants of the double-auction market; in this paper we analyse a clearing-house mechanism with uniform-pricing, meaning that: i) orders from all participants are queued up prior to the clearing operation, and ii) all trades take place at the same price (the mid-point of the market-quote as advertised prior to clearing).

III. Methodology

Since a traditional game-theoretic analysis of the double-auction is intractable [10], we analyse the double-auction market game using the *heuristic* methodology described in [11], [10]. The central idea is to restrict 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

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

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 VIII-A.

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.

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 the space of heuristic strategies used in our analysis.

IV. Heuristic Strategies

Each agent a_i has an associated trading strategy ζ_i, which specifies a mapping between its valuation v_i and the order that it will place at time t. For simplicity, we shall assume that: buyers always submit orders to buy (*bids*), sellers always submit orders to sell (*asks*), each agent only submits orders for a single unit; thus ζ merely specifies the *price* of the order according to the strategy being deployed.

We use three representative classes of strategy: truth-telling (TT), reinforcement-learning (RL) and Gjerstad-Dickhaut (GD) which are described in detail below. The

TT strategy was chosen since it is the simplest strategy that is able to achieve high efficiency outcomes in a homogeneous population in the clearing house 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 we included reinforcement-learning strategies since they are commonly used to model human game-playing behaviour in experimental economics [12].

A. The Truth-Telling Strategy

The truth-telling strategy (abbreviation TT) simply places orders equal to the agent's valuation: $\zeta(i,t) = v_i$.

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].

B. The Gjerstad-Dickhaut strategy

The Gjerstad-Dickhaut (abbreviation GD) [14] strategy uses historical data to estimate the probability of a order being accepted as a function of its price, and then chooses the price that maximises expected utility accordingly. This strategy is described in detail in [14].

C. 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.

These strategies choose their markup over their valuation price thus:

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

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

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: $RL_{\lambda_i} = \delta_{i_t}$, 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.

2) The Roth-Erev learning algorithm: The Roth-Erev algorithm (abbreviation RE) is designed to mimic human game-playing behaviour in extensive form games [12]. Agents bid probabilistically according to: $RL_{\lambda_i}(t) = RE_i(t) = \delta_{i_t}$ where $\delta_{i_t} \in \Theta_i$ is a discrete random variable distributed:

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

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

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

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

$$RE_q(\theta, a_i, t) = (1 - RE_{\rho_i})RE_q(\theta, a_i, t-1) + RE_\epsilon(\theta, a_i)$$

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

$$RE_\epsilon(\theta, a_i, t) = RL_{\rho_i}(t-1)[1 - RE_{\eta_i}] \iff \theta = RE_i(t-1)$$

$$RE_\epsilon(\theta, a_i, t) = RL_{\rho_i}(t-1)\frac{RE_{\eta_i}}{RL_{k_i}-1} \iff \theta \neq RE_i(t-1)$$

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

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

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

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

3) Nicolaisen et al.*'s modified Roth-Erev algorithm:* Nicolaisen, Petrov and Tesfatsion [8] (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:

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

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

$$RE_{\epsilon'}(\theta, a_i, t) = RL_{\rho_i}(t-1)[1 - RE_{\eta_i}] \iff \theta = RL_I(t-1)$$

$$RE_{\epsilon'}(\theta, a_i, t) = RE_{q_i}\frac{RE_{\eta_i}}{RL_{k_i}-1} \iff \theta \neq RE_i(t-1)$$

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 [15]. 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:

$$SQ_Q(\theta, a_i, t+1) = \\ SQ_Q(\theta, a_i, t) + SQ_{\alpha_i}[RL_{\rho_i} + \\ SQ_{\gamma_i} \max_{\theta'} SQ_Q(\theta', a_i, t) - SQ_Q(\theta, a_i, t)]$$

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:

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

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 $[0, RL_{k_i} - 1]$.

V. Objective Function

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 [10], we use *evolutionary* game-theory to model how agents might gradually adjust their strategies over time as they learn to improve their behavior in response to their payoffs. Thus we use the replicator dynamics equation [16]:

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

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 [2, p. 67].

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. 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:

$$T(\vec{x}, M \subset \triangle^n) = \\ |\{\vec{y} : \vec{y} \in M \wedge \vec{m}(0) = \vec{y} \wedge \exists t\ \vec{m}(t) = \vec{x} \wedge \dot{m}(t) = 0\}|$$

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 $\vec{x}$:

$$\beta(\vec{x}, M) = \frac{T(\vec{x}, M)}{|M|} \tag{1}$$

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:

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

Our method searches for strategies that are likely to be adopted under the replicator dynamics. More formally, we use an objective function that estimates the expected frequency with which our candidate strategy will be played in equilibrium outcomes. Thus our objective function is:

$$F(i, S, [H]) = \sum_{\vec{x} \in \epsilon_{[H]S}} \beta_{[H]}(\vec{x}, M) \cdot x_i \tag{2}$$

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 1 (p. 4), and $\epsilon_{[H]S}$ is the set of heuristic equilibria $\epsilon_{[H]S} = \{\vec{x} \in \triangle^{|S|} : \beta_{[H]}(\vec{x}, M) > 2 \times 10^{-2}\}$.

VI. Search Space

We use the objective function described in the previous section to search for strategies that are able to achieve high adoption rates under the replicator-dynamics model of social learning. We start with a population of agents able to use three different heuristic strategies: TT, GD, and RE which is an RL strategy that uses the Roth-Erev algorithm (section IV-C2) calibrated with parameters that best fit the data from human experiments [17], viz: $\forall i$ $RE_{k_i} = 50$ $RE_{\rho_i} = 0.1$ $RE_{\eta_i} = 0.2$ $RE_{s_i} = 9$ $RL_{\mu_i} = 1$.

Our goal is to search a space of strategies to acquire a new strategy that is likely to be adopted by this existing population. In previous work [18] a sensitivity analysis demonstrated that small perturbations in payoff estimates in favour of the RE strategy yielded subtantial improvements in basin-size for this strategy. This suggested generalisations of the RE strategy as possible candidates for further optimisation. RE belongs to the class of RL strategies, which we use as our search space. Thus in our experiment we have $S = \{\mathrm{s}^*, \mathrm{TT}, \mathrm{GD}, \mathrm{RE}\}$; s^* is a strategy represented as a 50-bit string, where:

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

VII. 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. Since we recompute all entries in the heuristic-payoff matrix in support of each candidate strategy, we use 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 due to sampling error thus 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. 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.

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* $\hat{\mathrm{OS}}$;
$[H]* \leftarrow \mathrm{GetHeuristicPayoffMatrix}(\mathrm{s}^* \cup S)$;
$\mathrm{OS} \leftarrow \arg\max_{\mathrm{s}^* \in \Pi} F(1, \mathrm{s}^* \cup S, [H]*)$;

Algorithm 1: FiSH

A GA was chosen 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.

As in [10], 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 [10]:

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

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. The 64-bit version of the Mersenne Twister random number generator [19] was used to draw all random values used

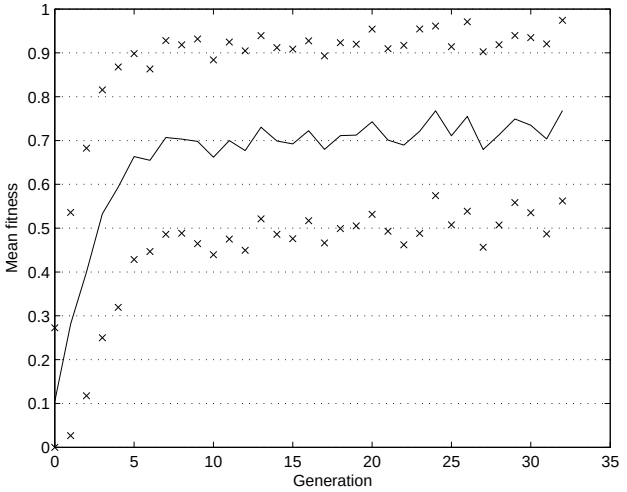

Fig. 1. Mean fitness of the GA population with one standard deviation

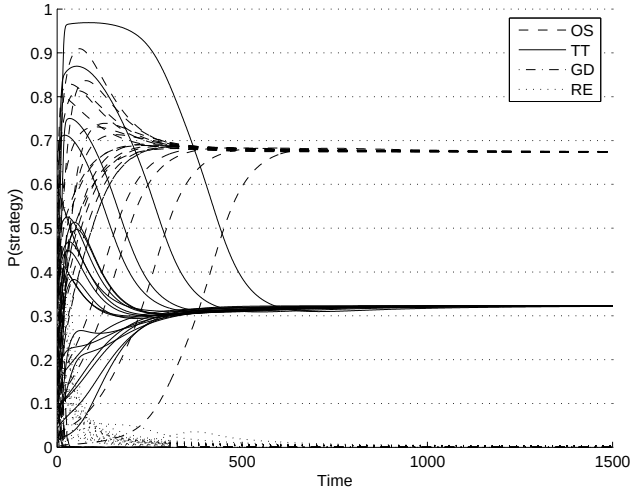

Fig. 2. Replicator dynamics time series plot for a 12-agent clearing-house auction showing interaction between optimised strategy (OS) versus GD, TT and the original Roth-Erev strategy (RE)

in the simulation. Each entry in the heuristic payoff matrix was computed by averaging the payoff to each strategy across 10^4 simulations.

VIII. Results

Figure 1 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 (denoted OS) that we evolved used the stateless Q-learning algorithm (SQ) with the following parameters: $RL_\mu = 1.210937$, $RL_k = 6$, $SQ_\epsilon = 0.18359375$, $SQ_\gamma = 0.4140625$ and $SQ_\alpha = 0.1875$.

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. Figure 2 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.

Taking $M \subset \triangle^4 : |M| = 10^3$ randomly sampled initial mixed-strategies, we calculate that there are two attractors: $\vec{A} = (0, 0, 1, 0)$ and $\vec{B} = (0.67, 0.32, 0, 0)$ 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.

A. An iterative approach

This method 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 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.

IX. Applications

Many algorithms for strategy-acquisition focus on searching for strategies that are generally *robust* when played against existing strategies. However, 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.

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 [6] 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 [20].

Secondly, the primary economic application of our method is to the *mechanism design* problem [21], [3]. In a mechanism design problem one attempts to define market "mechanisms", that is, the rules of the market, 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 IV-A) 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 [22]. 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 [6] 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.

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, however 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 2 to incorporate payoff maximisation in addition to basin size:

$$F'(i, S, [H]) = \sum_{\vec{x} \in \epsilon_{[H]S}} u(e_i, \vec{x}) \cdot \beta_{[H]}(\vec{x}, M) \cdot x_i \quad (3)$$

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.

X. Conclusion

We have introduced a novel method (algorithm 1) for acquisition of strategies in non-zero-sum n-player games. 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. Instead, our method focusses on searching for strategies that are *likely to be adopted*: we have formalised a metric (equation 2) for estimating the likelihood of adoption under a social learning process modelled by the replicator dynamics,

input : A set of heuristic strategies
$S = \{s_1, s_2, \ldots, s_n\}$ for some mechanism μ
output: A refined set of heuristic-strategies

$[H] \leftarrow \text{GetHeuristicPayoffMatrix}(S, \mu)$;
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**
      $\hat{F} \leftarrow F(i, S, [H]')$;
      $\text{i}^* \leftarrow i$;
      $\hat{\text{OS}} \leftarrow s_i$;
    end
  end
  if $\hat{F} < F(\text{i}^*, S, [H])$ **then return** S;
  $\Pi \leftarrow$ *create a search space based on generalisations of* $\hat{\text{OS}}$;
  $S' \leftarrow \text{s}^* \cup S$;
  $[H]' \leftarrow \text{GetHeuristicPayoffMatrix}(S', \mu)$;
  $\text{OS} \leftarrow \arg\max_{\text{s}^* \in \Pi} F(1, S', [H]')$;
  $S \leftarrow \text{OS} \cup S$;
  $[H] \leftarrow \text{GetHeuristicPayoffMatrix}(S, \mu)$;
  $S \leftarrow \text{EliminateDominatedStrategies}(S, [H])$;
until forever ;

Algorithm 2: FiSH+

based on an estimate of basin size (equation 1), and described how this can be calculated using numerical methods. We have validated our method empirically by applying it to a benchmark problem (sections VII–VIII).

Our method makes use of an evolutionary computing (EC) algorithm to perform heuristic optimisation. However it differs from existing EC methods for strategy-acquisition, such as co-evolutionary search in that we perform a full game-theoretic analysis over a small working set of heuristic strategies (section III), rather than a small sample of fitness comparisons over the full space of strategies.

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 from the National Science Foundation under grant NSF IIS-0329037 *Tools and Techniques for Automated Mechanism Design*.

REFERENCES

[1] 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: MIT Press, 1998.

[2] D. Fudenberg and D. K. Levine, *The Theory of Learning in Games*, 2nd ed. MIT Press, 1999.

[3] H. R. Varian, "Economic mechanism design for computerized agents," in *Proceedings of the First USENIX Workshop on Electronic Commerce*, 1995, pp. 13–21.

[4] J. Nash, "Equilibrium points in n-person games," in *Proceedings of the National Academy of Sciences*, vol. 36, 1950, pp. 48–49.

[5] D. Friedman and J. Rust, Eds., *The Double Auction Market: Institutions, Theories and Evidence*, ser. Santa Fe Institute Studies in the Sciences of Complexity. Cambridge, MA: Perseus Publishing, 1993.

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

[7] D. Friedman, "The double auction institution: A survey," in *The Double Auction Market: Institutions, Theories and Evidence*, ser. Santa Fe Institute Studies in the Sciences of Complexity, D. Friedman and J. Rust, Eds. Cambridge, MA: Perseus Publishing, 1993, ch. 1, pp. 3–25.

[8] 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*, vol. 5, no. 5, pp. 504–523, October 2001.

[9] 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*, vol. 24, pp. 17–27, 1998.

[10] 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.

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

[12] I. Erev and A. E. Roth, "Predicting how people play games: Reinforcement learning in experimental games with unique, mixed strategy equilibria," *American Economic Review*, vol. 88, no. 4, pp. 848–881, 1998.

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

[14] S. Gjerstad and J. Dickhaut, "Price formation in double auctions," *Games and Economic Behaviour*, vol. 22, pp. 1–19, 1998.

[15] J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, pp. 279–292, 1992.

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

[17] 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*, vol. 8, pp. 164–212, 1995.

[18] S. Phelps, P. McBurney, and S. Parsons, "A novel method for strategy acquisition and its application to a double-auction market game: full report," University of Liverpool, Tech. Rep. ULCS-09-019, October 2009.

[19] M. Matsumoto and T. Nishimura, "Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator," *ACM Transactions on Modeling and Computer Simulation*, vol. 8, no. 1, pp. 3–30, 1998.

[20] C. H. Papadimitriou, "Algorithms, games, and the internet," in *Proceedings of the 33rd Symposium on Theory of Computing*. ACM Press, 2001, pp. 749–753.

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

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