Evolutionary Mechanism Design

Thesis submitted in accordance with the requirements of the University of Liverpool for the degree of Doctor in Philosophy by Stephen George Phelps.

July 2007

MISSING PAGE/PAGES HAVE NO CONTENT

THE UNIVERSITY OF LIVERPOOL LIBRARY

THESIS USER'S DECLARATION

DATE	NAME (IN BLOCK LETTERS)	SIGNATURE	ADDRESS
14th January, 2009.			English Language Unit.

This thesis is dedicated to Margaret Phelps, who made it possible.

Acknowledgements

I am indebted to many people for their support and collaboration in the course of my Ph.D. research. Firstly I thank my supervisors Peter McBurney and Simon Parsons for their unwavering support and guidance over a long journey. A special thanks goes to Jinzhong Niu, Kai Cai and Marek Marcinkiewicz at the City University of New York for their collaboration on the JASA simulation software. In this respect, I would also like to thank Leigh Tesfatsion and Igor A. Walter for contributing bug reports, and Thierry Moyaux for his many contributions. I would also like to thank the many people who have contributed feedback and discussion to my research, amongst them, in no particular order: Michael Phelps, Andrew Byde, Enrico Gerding, Dave Cliff, Sam Hough, Alvin Roth, Tim Miller, Carles Sierra, Karl Tulys, Sieuwert van Otterloo, Seth Bullock, David Parkes, Elizabeth Sklar, Sevan Ficici, John Cartlidge, Omar Baqueiro Espinosa and William Walsh. I would like to especially thank Mike Wooldridge for his enduring patience, and Ruth Melville for her support and guidance beyond the call of duty.

My research was partially funded by EPSRC grant GR/T10671/01 - *Market Based Control of Complex Computational Systems: Techniques for Automated Mechanism Design*, and by the E.U. I.S.T. Programme through the SLIE project.

Contents

Glossary of Notations

$A = \{a_1, a_2, \ldots a_n\}$	The set of agents	p. 18
$B \subset A$	The set of buyers	p. 21
$S \subset A$	The set of sellers	p. 21
Ψ	The set of resources	p. 19
$\Omega : A \rightarrow 2^{\Psi}$	The ownership function	p. 19
$\Gamma : A \rightarrow \mathbb{R}$	The cash function	p. 20
$v_{a \in A} \in \mathbb{R}$	The true valuation of agent a	p. 21
$v_i \in \mathbb{R}$	The true valuation of agent a_i	p. 21
$\hat{v}_i \in \mathbb{R}$	The reported valuation of agent a_i	p. 26
$u_i : \Gamma \times \Omega \rightarrow \mathbb{R}$	The utility function for agent a_i	p. 20
$\chi_i : 2^{\Psi} \rightarrow \mathbb{R}$	The valuation function for agent a_i	p. 20
$\zeta(i,t)$	The shout price set by the strategy of agent a_i at time t	p. 39
R	The space of possible resource transfers	p. 19
C	The space of possible cash transfers	p. 19
P	The space of possible shouts	p. 28
$\text{pay} : 2^C \times \Gamma \rightarrow \Gamma$	The payment function	p. 20
$\text{trans} : 2^R \times \Omega \rightarrow \Omega$	The resource transfer function	p. 19
$VS \subset \mathbb{R}$	The supply schedule	p. 22
$VB \subset \mathbb{R}$	The demand schedule	p. 22
$MS \subset VS$	The efficient supply schedule	p. 23
$MB \subset VB$	The efficient demand schedule	p. 23
$MS' \subset VS$	The inefficient supply schedule	p. 24
$MB' \subset VB$	The inefficient demand schedule	p. 24
$\hat{MS} \subset P$	The efficient asks	p. 30
$\hat{MB} \subset P$	The efficient bids	p. 30
$\hat{MS'} \subset P$	The inefficient asks	p. 30
$\hat{MB'} \subset P$	The inefficient bids	p. 30
$E : C \rightarrow \mathbb{R}$	The gain from trade	p. 22
$TP \in \mathbb{R}$	The maximum possible gain from trade	p. 23
$EA : C \rightarrow [0,1]$	The efficiency of the market	p. 23
$p* \in [eq_a, eq_b]$	The equilibrium price range	p. 25
$(e\hat{q}_a, e\hat{q}_b)$	The market quote	p. 31
$K_t \subset A$	The set of active traders at time t	p. 29
E_t	The event history at time t	p. 29
$\triangle^n \subset \mathbb{R}^n$	The unit-simplex $\{\vec{x} \in \mathbb{R}^n : \sum_{i=1}^{n} x_i = 1\}$	p. 89
$\beta : \mathbb{R}^n \times 2^{\mathbb{R}^n} \rightarrow \mathbb{R}$	The basin size of an equilibrium in $\triangle^n$	p. 90

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

Much work in the design of multi-agent systems (MAS) [158] 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.

Such systems are known as *self-organising complex systems* (SOCS) [64] [1]. Examples of such systems are market places, ecosystems, nervous systems, neural networks, co-evolving systems, and of course, multi-agent systems. They are complex, in the sense that they consist of many parts with many interactions between them and exhibit non-linear, hard-to-predict behaviour, and they are self-organising in the sense that macro-level stabilities emerge despite the underlying complexity. As an example, consider a stock market consisting of hundreds of thousands of traders. Each trader

[1] The precise definition of a self-organising complex system is highly contentious, and there are many to choose from [55]. There is a particular sub-class of SOCS that exhibit a property called self-organised criticality [4], meaning that the attractors of the system lie on critical points (eg phase transitions) between order and chaos. It is suggested that the long-tail distribution of time intervals between events such as market crashes in the business cycle are due to criticality [81, 88]. However, for the purposes of this thesis the property of self-organised criticality is not considered the *essential* defining feature of a self-organising system or a market. Nevertheless the analysis and methods introduced in this thesis do not *preclude* chaotic dynamics or critically-poised behaviour. We will return to this discussion in Chapter 8.

is an autonomous agent, free to trade using whatever strategy they want. Individual prices at any given time are determined by the trading behaviour of all of other agents trading in the market; thus the actions of each agent can potentially influence all other agents; there are many interactions between the components of the system. Many aspects of the market's behaviour are chaotic or hard to predict, for example the price of an individual stock, or the profits of an individual trader. Yet despite this complexity, the variables that the stock-market "designer" is interested in, for example the overall market efficiency, remain at consistently satisficing values. Additionally, such systems are robust to exogenous perturbation; for example, after the stock market has been subjected to a shock, such as a market crash, the system eventually settles back into a state in which the design variables, for example market efficiency, are held at desirable values despite the fact that there is no explicit top-down control mechanism for achieving this. Such self-healing or homeostatic behaviour is typical of SOCS in general. These systems possess state-space dynamics with attractors and stable states (also known as equilibria) that lead the system to homeostatic states– that is, states in which our design variables are maximised or held within desirable ranges.

As designers of a multi-agent system, we are therefore tasked with ensuring that the complex system embodied by our MAS possesses attractors or equilibria in which our design objectives are met. But how can we affect the dynamics of our system if we are not allowed to prescribe the behaviour of individual agents — what free variables are at our disposal? The answer, of course is outlined above; in MAS design problems we typically have some control over the environment or infrastructure in which third-party agents interact. This can take the form of, for example, rules governing an auction mechanism, or the protocols used by agents for argumentation. Small changes in these rules or standards can have dramatic effects on the behaviour of the agents using these rules, and can radically alter the underlying dynamics of the system in surprising ways. By altering the underlying dynamics, we are sometimes able to adjust the system so that the stable states of the system exhibit the homeostatic properties we desire. For example, in a market-design context, by tweaking the rules of the market, we are sometimes able to design systems in which optimal allocative-efficiency is an emergent stable macro-property of the system.

Economists have studied similar design problems in the context of *auction theory* [80] and *mechanism design* [122, p. 640] [142]. In a mechanism design problem, the task of the designer is to choose the rules of the auction in such a way that the designer's objectives are met when agents play their optimal strategies. One of the main difficulties in solving this problem 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 — the set of strategies that are best responses to each other. For many scenarios, especially for single-sided auctions comprising a single seller and multiple buyers, auction theory and mechanism design yield clear-cut results. However, in the general case the problem is analytically intractable, especially when it comes to analysing *double-sided* auctions, also known as exchanges, in which we allow multiple sellers as well as multiple-buyers. In the next section I shall describe

our motivation for studying double-sided auctions.

1.1 Exchanges & their theoretical significance

A double-auction mechanism is a generalization of an auction in which there are multiple sellers as well as multiple buyers, and both buyers and sellers are allowed to exchange offers simultaneously. Since double-auctions allow dynamic pricing on both the supply side and the demand side of the marketplace, their study is of great importance, both to theoretical economists [77], and those seeking to implement real-world market places. On the one hand, economists who are interested in theories of price formation in idealized models of general markets have often turned to exchange-like models such as Walrasian tâtonnement, to describe and understand the price-formation process [17], and on the other hand, variants of the double-auction are used in large real-world exchanges to trade commodities in marketplaces where supply and demand fluctuate rapidly, such as markets for stocks, futures, and their derivatives.

However, the models of exchanges traditionally used by economists in general equilibrium theory are often simplified for the purposes of analytical tractability to such an extent that they are of scant relevance to the designers of real-world exchanges, and even, it is sometimes argued, of scant relevance to the theoretical modelling of markets [48]. For example, one important simplification often made is that the number of agents participating in a market is very large; this simplification allows relative market power and consequent *strategic effects* to be ignored. However, in many real-world marketplaces, such as deregulated wholesale electricity markets, there may be relatively few competitors on one or both sides of the market. With small numbers of participants, general equilibrium models break down [88, p. 10] because they fail to allow for market power, and the potential gains of strategic behavior, of participants.

1.2 Auction Theory & Mechanism Design

Auction theory can be thought of as an alternative approach to general equilibrium theory, in which we build a more sophisticated micro-model of the marketplace, and we use game-thoeretic techniques to analyse the rational behavior of *individual agents* faced with different pricing institutions. Whereas neoclassical equilibrium theory often abstracts away from the details of individual agents, game-theoretic models allow economists to build sophisticated micro-models of individual agents' reasoning and preferences. In many scenarios, especially in analyzing single-sided monopoly markets, these models have been spectacularly successful to the extent where they have been directly applied to the design of real-world auctions for high-value government and corporate assets [76]. However, in other practical scenarios, especially when it comes to analyzing and designing double-sided markets, such as exchanges, there are still a number of problems with the theory, which we shall briefly review.

Auction-theorists typically analyze a proposed market institution by defining a set of design objectives, and then proceed to show that these design objectives are brought about when rational agents follow their best strategies according to a game-theoretic

analysis. The task of choosing the rules of the market institution so that these objectives are brought about is called *mechanism design*. The typical design objectives considered by mechanism designers are[2]:

Allocative efficiency: The outcome of using the mechanism should be optimal in some defined sense, for example, the total surplus generated should equal the available surplus in competitive equilibrium.

Budget balance: No outside subsidy inwards or transfers outwards are required for a deal to be reached.

Individual rationality: The expected net benefit to each participant from using the mechanism should be no less than the net benefit of any alternative.

Incentive compatibility: Participants should not be able to gain an advantage from non-truth-telling behavior.

In many applications, auction-theory demonstrates the existence of market mechanisms that satisfy all of these properties when agents follow rationally prescribed bidding strategies. However, the impossibility result of [94] demonstrates that no *double-sided* auction mechanism can be simultaneously efficient, budget-balanced and individually-rational. Moreover, many of the underpinnings of the theory assume that designers' interests are restricted to only the aforementioned properties. For example, the revelation principle [80, p. 62] states that, without loss of generality, we may safely restrict attention to mechanisms in which agents reveal their types truthfully. However, this result does not take into account the potential cost or other practicalities of polling agents for their type information. Once minimizing the cost of revelation is introduced as a design objective, the revelation principle ceases to hold, because there may exist partial-revelation mechanisms with non-truthful equilibria which sacrifice incentive-compatibility for expedience of revelation. This is of more than academic interest, since in real-world electronic exchanges it is rarely possible to poll *all* agents for their valuations before clearing the market; hence the *continuous* double-auction, in which we execute the clearing operation as new offers arrive, thus increasing transaction throughput at the expense of incentive-compatibility.

In designing market places, as with any other engineering problem, we often need to make such tradeoffs between different objectives depending on the exact requirements and scenario at hand. We can often satisfactorily solve such multi-objective optimisation problems, provided that we have some kind of quantitative assessment of each objective, yet classical auction-theory provides only a binary yes or no indication of whether each of its limited design objectives is achievable, making it extremely difficult to compare the different trade-offs.

Further complications arise when we attempt to use auction-theory to analyze existing ("legacy") market institutions. Exchanges such as the London Stock Exchange[3] have been in existence far longer than game-theory and auction-theory, thus, unsurprisingly, the original rules of the institution were not necessarily based on sound game-theoretic or auction-theoretic principles. Moreover, it is unrealistic to expect that core

[2] I will give more formal definitions of these desiderata in Chapter 3

[3] `http://www.londonstockexchange.com/`

financial institutions such as these radically alter their rules overnight in response to the latest fashionable developments in auction-theory or game-theory. Rather, it may be more salient to view financial institutions *evolving* gradually and incrementally in response to a changing environment. Similarly, agents participating in these institutions do not necessarily instantaneously and simultaneously adjust their trading behavior to the theoretical optimum strategy; for example, adoption of a new trading strategy may spread through a population of traders as word of its efficacy diffuses in a manner akin to mimetic evolution.[4] Thus, we may think of the institutions we see today as the outcome of a *co-evolutionary* adaptation between financial institutions on the one hand, and trading strategies on the other.

The issue of legacy institutions has ramifications for mechanism design; in these contexts the choice of adjustments to the auction rules may be tightly constrained by existing infrastructure, both physical and social; thus it may be necessary to examine the *attainability* of equilibria under the new design given existing strategic behavior in the legacy design. Classical auction theory relies on classical game-theory which in turn says nothing about the *dynamics* of adjustment to equilibrium.

For such applications, we need to turn to models of evolution and learning in strategic environments; models that we collectively categorize under the banner of *evolutionary game theory*. Models of learning and evolution as applied to agents' strategies are not new. Where my approach differs, however, is in the application of models of learning and evolution to the market mechanism itself, a new field I call *evolutionary mechanism design*.

1.3 Thesis outline

In this thesis I introduce an iterative methodology for carrying out evolutionary mechanism design. The broad outline of the methodology is as follows. We start with an initial set of auction rules comprising a mechanism μ, in which we observe a set of trading strategies S. All of these are refined iteratively in response to direct observations of the real life marketplace (*in vivo* analysis), as well as forecasts based on simulation and game-theoretic analysis (*in vitro* analysis). The method is outlined by the pseudo-code on page 6: we start by performing an analysis of our initial strategies to see if there are hitherto unanalysed strategies that might upset the status quo (step 2); we then publicise our analysis to participants and update our analysis based on observations of the real market (steps 3 to 5); and finally we choose new mechanism rules that maximise our design objectives based on our current analysis of the market (step 7) before iterating the design cycle.

In the rest of this thesis I will define the various steps of this method in detail, and provide a empirical validation that it is both computable and effective. The outline is as follows. In Chapter 2, I survey the existing work that I draw upon. In Chapter 3, I define the space of mechanisms μ that will be analysed, and explore some of the difficulties that arise when using conventional *analytical* techniques to assess the properties of these mechanisms. In Chapter 4, I discuss the space of strategies S . Given

[4] The adoption by derivatives traders of the Black-Scholes equation for option pricing provides an example [84].

input : A set of initial heuristic strategies S, and a legacy mechanism μ
repeat
  $S \leftarrow$ `FiSH+` (S, μ) ;
  publicise S to participants;
  $\vec{x} \leftarrow$ *frequency of each strategy observed in vivo*;
  $S \leftarrow S \cup \{$ *strategies observed in vivo* $\}$;
  $\Lambda \leftarrow$ *space of feasible variants of* μ;
  $\mu \leftarrow \arg\max_{\mu* \in \Lambda}$ EvaluateDesignObjectives$(\mu*, S, \vec{x})$;
  implement rules defined by μ;
until forever ;

Algorithm 1: Evolutionary mechanism design

the difficulties of a purely analytical approach in assessing the properties of strategies and mechanisms, in Chapters 5 and 6 I introduce and validate a framework for *simulating* the interaction between strategies and mechanisms in order to assess the likely outcome. In Chapter 7, I give a brief overview of an existing methodology called empirical game-theory that can be used to combine the results of a simulation approach with a rigorous game-theoretic analysis. In Chapter 8, I introduce a semi-automated method for computing the function EvaluateDesignObjectives() using empirical game-theory in conjunction with the simulation framework. In Chapter 9, I introduce the algorithm `FiSH+`, which can be used to discover a *new* set of strategies that are likely to be played given a mechanism and an existing set of strategies. In Chapter 10, I outline a method for computing the full optimisation function $\arg\max_{\mu}$ EvaluateDesignObjectives() and empirically validate it with respect to a subset of the space of μ and S. Finally, in Chapter 11, I summarise my findings and discuss future work.

Chapter 2

Literature Review

2.1 Economics and Artificial Intelligence

It has long been understood that Artificial Intelligence (AI)[1] has strong roots in economics [122, p. 9]; whilst the latter is traditionally concerned with idealized models of agents[2] interacting in realistically complex environments, the former has placed more emphasis on realistically complex agents interacting in idealized environments. Indeed, one of the pioneers of AI, Herbert Simon[3] was originally motivated in much of his AI research by attempts to build more complex models of agents' behaviour in economic

[1]For the purposes of this chapter, the definition of AI is taken from [122, p. vii]: "The main unifying theme is the idea of an *intelligent agent*. We define AI as the study of agents that receive percepts from the environment and perform actions".

[2]Of course, the precise definition of the phrase "intelligent agent" is itself highly contentious. The use of the word "agent" in an AI context did not enter into mainstream use until the mid 1990s. However, taking the perspective of Russell and Norvig [122], this was not because intelligent agents did not exist prior to the introduction of this phrase, but rather because they were known by different terminology, and because the emphasis prior to the intelligent-agent approach was to work on the individual components of agent design (vision, planning, knowledge-representation, etc.) in isolation, without necessarily focusing on the inherent problems entailed in building a "whole-agent" architecture [122, p. 27]. However, researchers were still working on intelligent agents prior to 1995; whereas a planning system hooked up to a physical robot might have been called "an experiment in situated AI" during the 1980s, the same system might have been described as "an intelligent agent" in the late 1990s. Thus we will use the word agent to mean "an entity that receives percepts from the environment and performs actions. Each such agent implements a function that maps percept sequences to actions." [122, p. vii]. Since we will be sometimes be taking a decision-theoretic perspective, we will sometimes refer to this function as the agent's *decision function* (which solves its *decision problem*). Note that humans are are compatibile with this definition of an intelligent agent (since we take actions in our environment in response to sequences of percepts in accordance with some yet-to-be-formulated function), and we shall intentionally use the phrase intelligent agent ambiguously to refer to both artificial and "natural" agents; the latter tying in nicely with the usual meaning of the word agent in the economics literature (which predates its use in computer science [91]). However, I urge the reader not to take these definitions too rigidly; after all, to adapt a phrase from Shakespeare [128], an agent by any other name would act just as rationally.

[3]Herbert Simon was co-winner of the 1975 Turing prize for "basic contributions to artificial intelligence, the psychology of human cognition, and list processing" [97], as well as winner of the Nobel prize for economics in 1978, and was co-author of the first automated reasoning program [122, p. 17], the Logic Theorist, which was developed in 1955.

environments (see, for example [23]).

Whilst the broad relationships between the two disciplines were generally understood from the inception of AI, it was not until the late twentieth century and the birth of the Multi-Agent Systems (MAS)[4] [158] discipline that highly specialised theories and concepts were imported from economics into AI. Boutilier *et al.* [14] were amongst the first to clearly articulate the specific relationships between economics and AI. The particular significance of mechanism design in the context of multi-agent systems was first discussed in [117] and [142], 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. "* [151]

More recently the theme of incentive-engineering has been taken up in the wider computer-science community in contexts as diverse as information security [1], and computer networking:

> *"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.* All design problems are now mechanism design problems." [104]

2.2 The Double Auction

This thesis focuses specifically on a particular class of economic mechanism – the double auction. As discussed in the previous chapter, 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 [51] 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 this mechanism [123]. To this day the double-auction still represents an important benchmark problem by simultaneously admitting of precise representations whilst stretching the bounds of both analytical tractability and computational brute-force. In the following I will review analytical

[4]The field of MAS grew from distributed AI [37], and is principally concerned with the issues that arise when multiple intelligent agents interact with each other. This is in contrast to traditional AI which tended to focus on systems comprising a single agent. Multi-agent systems are generally harder to analyse because the outcome of one agent's action may depend on the action chosen by other agents.

and computational approaches to the agents' decision problem (traditionally the focus of AI), and the mechanism-design problem (traditionally the focus of economics) in turn.

2.2.1 Analytical approaches

The core of the analytic approach to agents' decision problems is based around the theory of n-player non-zero-sum games as formulated by John Nash [95], which I shall discuss in more detail in Chapter 7. Nash's insight was that in any interaction of preference-maximising agents whose outcome depends on the joint set of actions – that is, a game – any given agent has a theoretical *best response* to the actions chosen by the other agents. By applying this reasoning recursively we arrive at the concept of a Nash equilibrium; a situation in which every agent chooses actions that are best-responses to the best-responses of other agents. Nash proved that *every* n-player game possesses at least one equilibrium solution, thus providing a powerful theoretical framework not only for optimizing one's strategy in such an interaction (choosing a best-response), but also in predicting a likely combination of joint actions (Nash equilibrium). Many refinements have since been made to Nash's theory, some of the most important being Harsanyi's concept of a Bayesian-Nash equilibrium (BNE) [63], which deals with situations where payoffs are dependent on some private unobservable properties of an agent – the agent's *type* (for example, the particular cards that an agent holds in a game of poker), and Maynard Smith's theory of evolutionary games [86, 56] which overlays a dynamic model of gradual strategy-adjustment on top of the static equilibria of Nash's original formulation.

Game-theory provides a very powerful *general* framework for solving agent interactions in theory, but it was William Vickrey [143, 144] who first saw the fundamental economic significance of auctions and who first applied the theory of games in this area giving birth to modern auction theory, as summarised by Vijay Krishna in his comprehensive overview of the state of the art [80].

Auction theory provides a comprehensive theoretical framework for analysing single sided auctions – that is, auctions with a single seller and multiple buyers. However, double-sided auctions – auctions with multiple sellers as well as multiple buyers – remain something of a theoretical oddity despite their increasing prevalence in economic reality. Vickrey [143] demonstrated that no double-sided mechanism could simultaenously achieve the incentive-compatibility, individual-rationality, budget-balance and efficiency desiderata. Subsequently d'Aspremont and Gérard-Varet [32] demonstrated the existence of a budget-balanced mechanism that was able to achieve incentive-compatibility in Bayesian-Nash equilibrium[5] at the expense of individual-rationality. McAfee [87] provided a formulation of a double-sided single-unit mechanism that admitted of a dominant-strategy game-theoretic solution at the expense of budget-balance, and Huang *et al.* later refined this idea to the multi-unit case [67]. However Myerson and Satterthwaite [94, 127] were able to extend Vickrey's result and demonstrated that for the case of a single buyer and seller there does not exist a mechanism

[5]Bayesian-Nash incentive-compatibility merely requires truth-telling as Bayesian-Nash equilibrium of the game, rather than the usual stricter requirement that truth-telling is a dominant-strategy.

that can simultaneously achieve incentive-compatibility, budget-balance, efficiency and individual-rationality even when the incentive-compatibility criteria is relaxed from dominant-strategy to BNE, and hence there is no double-sided mechanism for achieving all the usual desiderata required by auction theorists in the general case.

Although there is no unequivocal and complete game-theoretic analysis of the double-auction in the general case, that is not to say, however, that double-sided mechanisms do not admit of game-theoretic solutions in specific instances. The first equilibrium analysis for a double auction was that of Chatterjee and Samuelson [21], in the paper in which they introduced the idea of the k-double auction[6], which we will discuss in the next chapter, albeit only for the two trader case. In this initial paper, Chatterjee and Samuelson show that there is an equilibrium solution, assuming independent private values.

Considerable work has since been carried out extending this result. First, Williams showed the existence of equilibria in the buyer's bid double auction [155, 154] — this is an easier auction to analyse since the dominant strategy for sellers is to bid their true value, thus fixing one side of the auction and, as [124] points out, ensuring that the market has a unique equilibrium[7]. The same authors subsequently showed the existence of equilibria in the many-trader version of the k-double auction [127], at the same time suggesting that the modified BBDA has no equilibrium. This work was followed by Jackson and Swinkels [70, 71], who showed the existence of equilibria, though not monotonic equilibria, under a wide range of conditions. Next, Reny and Perry [115] showed that monotonic equilibria exist if offers are restricted to discrete values, and Fudenberg *et al.* [53] showed that this result could be extended to continuous values (which [71] argues is "a very useful approximation ... allowing one ... to use calculus to characterise equilibria") provided that the auction was large. Finally, Kadan [73] showed that an increasing equilibrium exists for just two traders with affiliated values.

2.2.2 Empirical approaches

Whilst double-auction mechanisms stretch the bounds auction-theory by admitting of no unequivocal dominant strategy solution in the general case, the theory of games itself has come under scrutiny as a plausible general-purpose model of the strategic behavior of complex agents (human or otherwise); for example, Goeree and Holt [59] give an overview of ten simple games where the game-theoretic solution is easily obtainable yet intuitively implausible. This has led to a re-examination of the use of *empirical* methods in economics, whereby experiments are conducted with actual agents trading in a market-institution under laboratory conditions. The agents may be human, in which case the methodology is sometimes called *experimental economics* (see for example [75]), or more generally they may be implemented in the form of a computer-program; Tesfatsion [139] coined the phrase *agent-based computational economics* (ACE), to describe this approach.

[6] Though not under this name — they refer to the price setting rule as a "bargaining rule".

[7] Note that all results for the BBDA, the 1-DA, are symmetric with those for the k-double auction in which the transaction price is determined by the price offered by the highest asking seller that trades, the 0-DA [124].

Experimental economics using human agents has the advantage that a large supply of agents are available "off the shelf" so to speak; hence not surprisingly experiments using human agents were among the first ACE investigations of the double-auction market. Smith [131] was the first to study the double-auction under laboratory conditions using human-agents, and his results suggested that human subjects were able to extract close to theoretically optimal surplus from the market.

One of the disadvantages of human-based experimental economics compared with agent-based computational economics is that it is not always straightforward to analyze the necessary cognitive mechanisms required to achieve a particular economic outcome. In contrast, Gode and Sunder [58] performed one of the earliest agent-based experiments on the double-auction with the aim of investigating the lower-bounds on the amount of cognitive machinery required to achieve efficient outcomes. They were able to demonstrate that their minimal *zero-intelligence* strategies, implemented in the form of computer programs, were able to achieve highly efficient outcomes, suggesting that the double-auction mechanism was highly robust in the sense that it required minimal rationality on behalf of participants. Their results were not unequivocal, however; Cliff and Bruten [28] demonstrated that some aspects of Gode and Sunder's results were highly contingent on the particular distribution of agents' valuations that were used in the original experiments, and that a more sophisticated and robust strategy, *zero-intelligence plus* (ZIP) was required in order more accurately fit the behaviour of human subjects under less restrictive assumptions.

This was not the end of the story, though, since when analysing a market mechanism ideally we want to demonstrate the existence of a *dominant* strategy, and that design objectives such as high-efficiency outcomes are the result of agents adopting this particular strategy. For example, in many single-sided auctions one of the desiderata usually considered is *incentive-compatibility*; the dominant bidding strategy should be to bid truthfully at one's valuation. Unless we can demonstrate that an economic outcome such as high efficiency is the result of agents adopting a dominant strategy, or at the very least an equilibrium strategy profile, we can never be sure that the strategy under which high efficiency is observed will not, at some point, be discarded in favour of an alternative strategy which yields higher payoff for its adopters at the expense of overall social welfare. By analogy, consider the prisoner's dilemma game [49, 9, 3]; although the cooperative strategy yields the highest welfare outcome if all agents adopt it, this does not suffice to demonstrate that both agents will adopt the cooperative strategy since there will always be a temptation to choose the defection strategy.

Thus there have been numerous attempts to craft agent-based trading-strategies for double-auctions that are able to out-compete other strategies: Preist and van Tol [112] devised a variant of Cliff's ZIP strategy that was able to trade in persistent-shout auctions; Gjerstad and Dickhaut (GD) introduced a trading strategy that estimates the probability of a bid being accepted as a function of bid price based on an analysis of historical market data, and then bids to maximise expected profit [57]; Todd Kaplan's [51] entry into the Santa Fe tournament was one of the first documented double-auction *sniping* strategies, which wait until the last minute before submitting a bid in order to prevent counter-bidding; Tesauro and Das [138] introduced variants of the GD and ZIP strategies that were able to trade in continuous-time environments; Nicolaisen et al. [98] used a trading strategy based on Roth and Erev's [43] reinforcement-learning [74]

model of human game playing to analyse a simulated electricity market; and Hsu and Soo [66] analysed the performance of a strategy based on the q-learning algorithm [147]. Variations on these and other strategies have been pitted against each other in several public tournaments designed to elicit new strategy designs from the ACE community [61, 153, 123]. Some of these strategies will be discussed in full detail in Chapter 4. Although some of them have advantages over others in certain situations, and there are pros and cons to each, there is evidence to suggest that none of them are *dominant* over the others [145], even putting aside the problem of demonstrating that any are dominant over the entire space of possible strategies.

Evolutionary search

Much of the work cited in the previous section focussed on showing that particular strategies yield high payoff if deployed in a market in which all agents adopt the same strategy homogenously. However, if we have reason to believe that none of the strategies from the previous section are dominant over the others when they iteract with each other in the same marketplace, we have no reason to believe that any *single* one of them will come to be used in a real market. Hence if we simply compute market outcomes by running experiments in which we equip agents homogeneously with the same non-dominant strategy, we are not necessarily nearer to understanding the economic properties of the double-auction.

Of course, it may be the case that a single dominant strategy simply does not exist for the double-auction game; instead, some *mixture* of these, or yet to be discovered strategies, might constitute a Nash *equilibrium*. That is, even though no single strategy is "optimal" in the sense that it is dominant over the others, some mix of these or other strategies might constitute best-responses to each other. If this were the case and our market were populated by such a mix of strategies, we might expect that such a state of affairs would persist in reality, since by definition if the agents were to change their strategy they would be worse off. Therefore the agents themselves would have an incentive to maintain the status quo; and thus the components of the system would tend to naturally drive the system back towards such an equilibrium. Thus if we evaluate the properties of the mechanism when it is in these equilibrium states, we might expect that our predictions for variables such as market-efficiency will be accurate for some reasonable duration, and if our design objectives are maximised in these equilibria we will have shown that our mechanism is homeostatic.

In order to assess whether or not there are mixtures of strategies constituting equilibria, it is necessary to systematically evaluate the strategic *interaction* between the known strategies, as well as the space of yet to be considered strategies. Since this search-space is very large, exhaustive search is unfeasible. This has led researchers to turn to heuristic methods such as evolutionary search as a possible methods for studying the interaction between different double-auction strategies by systematically sampling the search space, e.g.: Cliff [25] used evolutionary search to explore the parameter space of his ZIP strategy, and Andrews and Prager used Koza's genetic programming technique [79] to search for a best-response to a uniform mixed-strategy of the Santa Fe tournament entries. *Co*-evolutionary algorithms [65, 2, 111] are highly promising in this respect. In a co-evolutionary search the fitness of individuals in the popula-

tion is evaluated relative to one another in joint interactions (similarly to payoffs in a strategic game), and it is suggested that in certain circumstances the converged population is an approximate Nash solution to the underlying game; that is, the stable states, or equilibria, of the co-evolutionary process are related to the game-theoretic equilibria. Price [113] and Dawid [34] used co-evolutionary search to explore convergence to equilibrium states in the double-auction.

However, there are many caveats to interpreting the equilibrium states of standard co-evolutionary algorithms as approximations of game-theoretic equilibria, as discussed in detail by Sevan Ficici [46, 45]. This has led to a number of refinements to standard co-evolutionary algorithms by incorporating game-theoretic concepts directly into the co-evolutionary algorithm itself [100, 47, 44]; the use of heuristic search (evolutionary or otherwise) to find approximate best-response or equilibrium strategies is an open research topic that I shall return to in Chapter 9.

2.2.3 A hybrid approach: empirical game-theory

The various caveats discussed above with the game-theoretic, agent-based and evolutionary approaches, as used in isolation, have inspired *hybrid* approaches whereby agent-based experimentation is used to build an approximate game-theoretic representation which is then solved using standard techniques from classical and evolutionary game-theory. This methodology is known as *empirical* game-theory, and it is the principle methodology used in this thesis, as described in Chapter 7. Many studies prior to 2000 had started to take a more principled and systematic approach to studying the interaction between complex strategies in a simulation context, for example Rust, Miller and Palmer systematically studied convergence to equilibrium of the strategies in the original Santa Fe tournament using ideas very similar to evolutionary game-theory [51, p. 183–189]. These ideas matured within the MAS community, and a research group at Michigan set this kind of analysis in a rigorous game-theoretic terms: in 2002 Walsh et al. demonstrated the effectiveness of the technique for several bargaining games, including a double-auction [145]; Walsh, Parkes and Das introduced a refinement to the technique to concentrate the sampling of simulations on those experiments that were most critical to the equilibrium analysis [146]; Reeves et al. performed a game-theoretic analysis of strategies in a market-based scheduling scenario [30] and Wellman et al. [152] used empirical game-theoretic analysis to help design their entrant on the 2004 trading agent competition.

2.3 Automated mechanism design

Whilst the application of computational techniques to the agent decision problem has a comparatively long tradition, their application to the mechanism-design problem is more recent. The economist Alvin Roth was the first to pose mechanism-design as an *engineering* problem [118], thus paving the way for the application of engineering techniques to mechanism-design. Cliff [26] and myself [108, 107] were the first to apply ad-hoc evolutionary search to the double-auction design problem with a view to automating the mechanism-design process (I present my earlier work in this area in

Chapter 10). Meanwhile Connitzer and Sandholm [29, 126] were the first to pose the automated mechanism-design problem in rigorous theoretical terms and analyze the algorithmic complexity of the problem. Byde [19] used computational techniques to analyze a space of variants to the Vickrey nth-pricing rule in the context of single-sided auctions, and David *et al.* used Bayesian learning to optimize the rules of a single-sided auction mechanism in cases where agents are constrained to discrete bid prices [33].

2.4 Evolutionary mechanism design

The central theme of this thesis is that just as choice of strategy is not a static problem, since agents may be constrained in their adjustment of strategy over time, neither is mechanism-design; mechanism designers may also be constrained in their choice of mechanism rules, for example there may be legacy infrastructure that prevents an institution such as a large stock exchange from radically altering its auction rules overnight. Just as constraints on strategy adjustment lead to *evolutionary* game theory, constraints on mechanism adjustment lead to *evolutionary* mechanism-design. We might think of the market institutions that we observe today as the equilibrium outcome of a co-evolutionary process not just between individual strategies, but a coevolution between strategy and mechanism. Peyton Young was the first economist to propose this idea [161], and it is a theme I shall revisit in Chapters 8 and 10.

2.5 Summary and Contribution

Economists have long used idealized models of agent behaviour in order to understand market behaviour. AI practitioners have had to adapt these models in order to build actual agents, and the resulting engineering approach to agents' behaviour requires more sophisticated and complex models. Similarly, it has recently been understood that the idealized notion of a "free" market is not always applicable, since actual markets entail many rules that govern their operation. Building real markets entails an engineering approach just as does the building of real agents.

In this thesis I introduce several engineering methods for *evolutionary* mechanism design in the context of double-auction markets. In Chapter 8 I discuss an application of empirical game-theory to analysing different pricing rules for a double-auction with particular emphasis on the applicability of this technique for *legacy* mechanism design. This work first appeared in [106]. In Chapter 9 I introduce a novel method for automated strategy acquisition that can be used as a method for *intervening* in an existing mechanism in order to perturb the equilibrium of the system back into a socially desirable state. This work was originally presented in [105]. Finally, in Chapter 10 I present one of the first attempts to use evolutionary algorithms to directly search the mechanism-design space, which was originally presented in [109].

The following is a list of my refereed publications that were published during the course of the research that I conducted for this thesis:

- [105] S. Phelps, M. Marcinkiewicz, S. Parsons and P. McBurney. A novel method for automatic strategy acquisition in n-player non-zero-sum games. In

H. Nakashima, M. P. Wellman, G. Weiss and P. Stone, editors, *Proceedings of the 5th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2006)*, pages 705–712, Hakodate, Japan, May 2006. ACM.

- [106] 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.

- [107] S. Phelps, S. Parsons, P. McBurney, and E. Sklar. Co-evolution of auction mechanisms and trading strategies: Towards a novel approach to microeconomic design. In *Proceedings of the Bird of a Feather Workshops, Genetic and Evolutionary Computation Conference*, pages 65–72, New York, July 2002. AAAI.

- [108] S. Phelps, S. Parsons, P. McBurney and E. Sklar. Co-evolutionary mechanism design: A preliminary report. In J. Padget, O. Shehory, D. Parkes, N. Sadeh, and W. E. Walsh, editors, *Agent-Mediated Electronic Commerce IV: Designing Mechanisms and Systems*, pages 123–143, Springer Verlag, July 2002.

- [109] S. Phelps, S. Parsons, E. Sklar and P. McBurney. Using genetic programming to optimise pricing rules for a double auction market. In *Proceedings of the workshop on Agents for Electronic Commerce*, Puitsburgh, PA, October 2003.

- [110] S. Phelps, V. Tamma, M. Wooldridge and I. Dickinson. Toward Open Negotiation. *IEEE Internet Computing*, 8:(70–76), 2004.

Chapter 3

A Generic Model of the Double-Auction

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.

In this chapter I shall describe in detail the operation of this type of marketplace. However, arriving at a comprehensive description that is rigorous enough for formal analysis is a difficult task. Many variants of this institution exist in the real-world, and hence similarly in the economics literature. The differences between these variants can be subtle and hard to describe since the trading rules governing real-world exchanges have evolved over many decades, in many different countries. Hence there are no definitive standards or terminology for formal modelling of these institutions.

There have been several attempts at formally defining a general space of possible auction mechanisms, and modelling double-auction variants as points within this space [160]. I shall take a more constrained approach, however, since

- These approaches attempt to provide a general framework for classifying all types of auction mechanism, not just double-auctions, and hence these models have a great many parameters. By adopting a less general model, we expect to be able to build a simpler framework with fewer parameters that will be more tractable for my purposes

- Any model is necessarily an *abstraction* of some real-world phenomenon. Abstraction involves discarding details that are felt to be irrelevant for the purpose at hand, and the models thus obtained incorporate many assumptions about what is relevant and what is not. However, what is relevant can vary significantly from problem to problem. This is especially the case when we are analysing *artifacts*

which do not share a single designer or design process, and when our purposes are practical in nature, such as when our problem is a design problem. Both of these hold in the analysis of auction mechanisms. For example, the *standard* theoretical model of an English ascending auction assumes that auctions are short in duration, and that there is no opportunity cost to bidders in placing bids or monitoring the auction. These assumptions hold in bricks-and-mortar auctions, but fail to hold in many internet auctions, and thus alternative models are required [82]. This is a reflection of the fact that many problems in economics are engineering problems [118], and thus as with other engineering disciplines, for example, software engineering, we should expect our models to be highly project-specific and somewhat disposable in nature.

Bearing in mind these considerations, we review several different double-auction institutions that are commonly discussed in the literature. We compare and contrast their differences from a design perspective, and proceed to construct a model that encompasses each variation as a special case whilst capturing the design-relevant differences between each institution. This model will then be used throughout the thesis to illustrate different economic design methodologies. As befits an engineering approach, we will use a number of different modelling languages to illustrate our framework, including the Universal Modeling Language (UML) [121], which is commonly used by software engineerings not only to model software systems, but also the wider extra-computer environment in which software systems are embedded.

Since our model does not attempt to be all-encompassing, this introduces some caveats. Firstly, we cannot make claims about all possible auction variants, such as claiming that a particular mechanism is *the* optimal one with respect to a given set of design objectives. Secondly, we cannot provide an *apriori* guarantee that our methods are applicable under alternative models.

However, as we reasoned earlier, in most real-world problems these caveats are also applicable to so called general models, since we will always be able to find a scenario that violates certain of the assumptions of any given theory. Throughout this chapter, we will see that many real-world double-sided mechanisms violate some of the fundamental assumptions of auction theory, such as the revelation principle, and are thus outside the space of mechanisms traditionally considered by auction theorists.

Rather than attempting to circumvent these caveats, we will instead adopt an engineering approach; our discourse will not encompass the theoretically possible, rather it will be limited to relevant design characteristics of interest; when we introduce design methodologies, we will take a heuristic approach, and talk about *good*, rather than *optimal* designs.

3.1 A model of a commodity-exchange market

3.1.1 The resource allocation problem

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 class of resource Ψ is traded in the market place. The resource is divided up into *units*: $\Psi = \{\psi_1, \psi_2, \dots\}$. Each individual unit of the resource is indivisible.

Each trader *owns* a certain subset of the resource Ψ defined by the function

$$\Omega : A \rightarrow 2^{\Psi}$$

where $\Omega(a_i) \subset \Psi$ denotes the units of resource to which trader a_i has exclusive access, and with which it is free to do with as it pleases.

The resource is *non-sharable*; that is:

$$\Omega(a_i) \cap \Omega(a_j) = \emptyset \; \forall_{i \neq j}(a_i, a_j) \in A^2$$

The function Ω defines the *allocation* of the resource Ψ amongst the traders A. Traders cannot be coerced into relinquishing ownership of resources, but they may volunteer to *transfer* a certain number of units of resource to another trader which results in a new allocation. A transaction involving the resource is represented by a tuple $r = (r_i \in A, r_j \in A, r_\psi \in 2^{\Psi}) \in R$ representing a transfer of r_ψ units from trader r_i to trader r_j. The function mapping from an original allocation Ω to the allocation resulting from a transaction $r \in R$ is:

$$\Omega' = \text{trans}(\{r\}, \Omega)$$

where:

$$\begin{aligned} \Omega'(a_j) &= \Omega(a_j) \cup T \\ \Omega'(a_i) &= \Omega(a_i) - T \\ \Omega'(a_x) &= \Omega(a_x) \forall_{x \neq i \neq j} a_x \in A \end{aligned}$$

For multiple transactions the trans function is defined recursively. Given a set of transactions $RS \subset R = \{rs_1, rs_2, \dots, rs_k\}$, and an initial allocation Ω, the allocation resulting from the sequence of transactions in RS is given by

$$|RS| > 1 \implies \Omega' = \text{trans}(RS, \Omega)$$

where:

$$\begin{aligned} \omega_0 &= \Omega \\ \Omega' &= \omega_k \\ \omega_i &= \text{trans}(\{rs_i\}, \omega_{i-1}) \; \forall rs_i \in RS \end{aligned}$$

Traders participate in the market in order to exchange units of Ψ for *cash*. The amount of cash owned by an trader is given by the function $\Gamma : A \rightarrow \mathbb{R}$. Traders cannot be coerced into relinquishing cash, but they may volunteer to transfer a certain amount of cash to another trader, which again results in a new allocation. A transfer of cash is represented by a tuple $c = (c_i \in A, c_j \in A, c_p \in \mathbb{R})$ meaning that trader c_i transfers

c_p to trader c_j. The function pay maps from an original cash allocation Γ to the new allocation Γ' resulting from a cash transfer thus:

$$\Gamma' = \text{pay}(\{c\}, \Gamma)$$

where:

$$\begin{aligned} \Gamma'(c_j) &= \Gamma(c_j) + c_p \\ \Gamma'(c_i) &= \Gamma(c_i) - c_p \end{aligned}$$

For multiple transactions the pay function is defined recursively as per the trans function.

Typically, traders enter in mutual transfers of cash and resource. If a trader a_i transfers cash to trader a_j, and in return trader a_j transfers resource to trader a_i, then we say that a_i *buys* resource, and that trader a_j *sells* resource.

Each trader a_i has different *preferences* over the possible allocations of cash Γ and resource Ω. Preferences are defined by the trader's *utility function*:

$$u_i(\Gamma, \Omega) = u(a_i, \Gamma, \Omega) \tag{3.1}$$

A trader i *prefers* an allocation (Γ', Ω') over an alternative allocation (Γ, Ω) if, and only if:

$$u_i(\Gamma', \Omega') > u_i(\Gamma, \Omega)$$

A trader i is *indifferent* over two allocations (Γ', Ω') and (Γ, Ω) if, and only if:

$$u_i(\Gamma', \Omega') = u_i(\Gamma, \Omega)$$

In the scenarios that we shall study, Ψ is a *commodity*; that is, traders are indifferent over allocations in which they own the same number of items of Ψ. More formally:

$$|\Omega_x(a_i)| = |\Omega_y(a_i)| \implies u_i(\Gamma, \Omega_x) = u_i(\Gamma, \Omega_y)$$

We shall also assume that traders' preferences are solely determined by their own allocations of resource and cash and not by those of other agents; that traders always prefer to have the greater of two bundles of cash; and that each trader i has a *valuation function* $\chi_i : 2^\Psi \to \mathbb{R}$ for their current allocation of their resource meaning that:

$$\begin{aligned} (\Omega'(a_i) = \Omega(a_i) - \psi_x) \wedge (\Gamma'(a_i) = \Gamma(a_i) + \chi_i(\psi_x)) & \\ \implies u_i(\Gamma', \Omega') &= u_i(\Gamma, \Omega) \end{aligned}$$

Accordingly, in our particular model, each trader's utility is given by a function of the form:

$$u_i(\Gamma, \Omega) = \Gamma_i + \chi_i(\Omega(a_i))$$

Our model of utility is further simplified by dividing traders 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 $S \cap B = \emptyset$. Our valuation function is then:

$$\begin{aligned} \Omega(a_i) = \emptyset &\implies \chi_i(\Omega(a_i)) = 0 \\ |\Omega(a_i)| > 0 \wedge a_i \in B &\implies \chi_i(\Omega(a_i)) = v_i \\ a_i \in S &\implies \chi_i(\Omega(a_i)) = v_i|\Omega(a_i)| \end{aligned}$$

where $v_i \in \mathbb{R}$ is the valuation of agent i for a single unit of resource.

Buyers can *cash in* their allocation of resource. If buyer $b_i \in B$ cashes in, then

$$\begin{aligned} \Omega_{t+1}(b_i) &= \emptyset \\ \Gamma_{t+1}(b_i) &= \Gamma_t(b_i) + v_i \end{aligned}$$

Sellers can *produce* additional resource. If seller $s_i \in S$ produces a single unit of resource $\psi_x \in \Psi$ then

$$\begin{aligned} \Omega_{t+1}(s_i) &= \Omega_t(s_i) \cup \psi_x \\ \Gamma_{t+1}(s_i) &= \Gamma_t(s_i) - v_i \\ \Psi_{t+1} &= \Psi_t \cup \psi_x \end{aligned}$$

In general, traders will only perform actions that increase their own utility. We will refer to such actions as *individually-rational* actions.

Note that since, in the general case

$$(\exists b_i)_B(\exists s_j)_S \; v_i > v_j$$

there may exist the possibility for traders to increase their utility by entering into mutual transfers of cash and resource. That is, in general, there are potential *gains from trade.*

3.1.2 Optimal allocations and the equilibrium price

A natural question then is how we can maximise the utility of all agents by selecting a set of transactions of cash and resource that are individually-rational for individual agents. More formally, given an initial allocation (Γ, Ω), we need to solve the following optimization problem:

$$\arg\max_{(C*,R*)} \sum_{i=1}^{|A|} u_i(\text{pay}(C*, \Gamma), \text{trans}(R*, \Omega))$$

We restrict attention to scenarios in which sellers produce resource which they then sell to buyers. Accordingly, for each tuple $c \in C*$

$$c_i \in B$$
$$c_j \in S$$
$$c_p \in \mathbb{R}$$

and

$$\forall_{c \in C*} \exists_{r \in R*}\ r_i = c_j \wedge r_j = c_i \tag{3.2}$$
$$\forall_{r \in R*} \exists_{c \in C*}\ c_i = r_j \wedge c_j = r_i \tag{3.3}$$

Let $v_b(c)$ denote the valuation of the buyer involved in the transaction, and let $v_s(c)$ denote the valuation of the corresponding seller:

$$\begin{aligned} v_b(c) &= v_{c_i} \\ v_s(c) &= v_{c_j} \end{aligned}$$

Assuming $v_s(c) < c_p < v_b(c)$, the gain in utility to each trader involved in a transaction c is $v_b(c) - c_p$ for the buyer, and $c_p - v_s(c)$ for the seller. Therefore, the total gain from trade for a solution $C*$ is:

$$E(C*) = \sum_{c \in C*} v_b(c) - v_s(c) \tag{3.4}$$

We can solve this maximisation problem by choosing the elements of $C*$ so that buyers with higher valuations are paired with sellers with lower valuations. Let the function $V : 2^A \rightarrow 2^{\mathbb{R}}$ denote the multiset of valuations corresponding to a given set of traders:

$$V(T) = \{v_i : a_i \in T\}$$

Let $VB = \{vb_1, vb_2, \dots\}$ denote the multiset $V(B)$, where vb_1 denotes the highest valuation of any buyer, and vb_i denotes the i^{th} highest valuation of any buyer. So that we have

$$\forall_{ij}\ i < j \implies vb_i \geq vb_j$$

Similarly, let $VS = \{vs_1, vs_2, \dots\}$ denote the multiset $V(S)$ where, where vs_1 denotes the *lowest* valuation of any seller, and vs_i denotes the i^{th} lowest valuation of any seller.

VS is called the *supply schedule*, and VB is the *demand schedule*. These have corresponding natural graphical representations which, in the continuous case (eg $VB = [a, b]$ where a and b are arbitrarily constants $\in \mathbb{R}$), can be represented as smooth curves known as the supply and demand curves. We retain this nomenclature for the discrete graphical representation of supply and demand.

Let MB and MS denote the subsets of VB and VS where buyer valuations *match* seller valuations; that is, where buyer valuations are greater than seller valuations:

$$\begin{aligned} MB &= \{mb_1, mb_2, \ldots\} \subset VB \\ MS &= \{ms_1, ms_2, \ldots\} \subset VS \end{aligned}$$

such that:

$$\begin{aligned} mb_i &\geq ms_i \; \forall i \\ mb_1 &\geq mb_2 \geq mb_3 \geq \ldots \\ ms_1 &\leq ms_2 \leq ms_3 \leq \ldots \end{aligned}$$

Claim 3.1 The maximum possible gain from trade is:

$$TP = \sum_{i=1}^{|MB|} mb_i - ms_i \tag{3.5}$$

Proof. We will prove this claim using a Reductio ad Absurdum argument.

Let b_i denote the buyer whose valuation is vb_i and let s_i denote the seller whose valuation is vs_i.

Suppose that the optimal gain from trade can be obtained through a set of transactions $C*$ involving at least one transaction involving a pair of traders b_i and s_j where $i \neq j$. Then equation 3.4 will contain a term

$$mb_i - ms_j$$

However, if $i < j$, then we could obtain a larger value of E, since we could choose a set of transactions C' in which we pair σ_i with b_i, instead of b_i and s_j and

$$\begin{aligned} i < j &\implies ms_i < ms_j \\ &\implies E(C') > E(C*) \end{aligned}$$

This contradicts our original assertion that $C*$ is optimal, and thus the result holds by Reductio ad Absurdem.

□

The ratio

$$EA(C) = \frac{E(C)}{TP} \tag{3.6}$$

is known as the *efficiency* of the market. The market is *efficient* if, and only if, $EA = 1$.

The Equilibrium Price

Of particular interest are solutions to the maximisation problem in which all transactions share a common price $p*$ so that we have $(\forall c)_{C*}\ p(c) = p*$. Faced with any given price p, any given buyer $b_i \in B$ will voluntarily buy from any seller $s_j \in S$ at the specified price provided that $p \leq v_i$, otherwise they will refrain from entering into a transaction. Similarly, any given seller $s_i \in S$ will voluntarily sell to any buyer $b_j \in B$ at the specified price provided that $p \geq v_i$. Thus given any p our transaction set C consists of all transactions satisfying the following constraint:

$$C = \{(a_i, a_j, p) : a_i \in S \wedge a_j \in B \wedge v_i \leq p \leq v_j\}$$

The total increase in utility across all traders is thus given by:

$$\begin{aligned} S(p) &= \sum_{a_i \in S \wedge p > v_i} p - v_i \ + \sum_{a_i \in B \wedge p < v_i} v_i - p \\ &= \sum_{a_i \in B \wedge a_j \in S \wedge p \leq v_i \wedge p \geq v_i} v_i - v_j \end{aligned} \tag{3.7}$$

We refer to this metric as the *social welfare* of the market, and our maximisation problem is

$$\arg\max_{p*} S(p*)$$

We can solve

$$S(p*) = TP \tag{3.8}$$

from equations 3.7 and 3.5:

$$\sum_{a_i \in B \wedge a_j \in S \wedge p* \leq v_i \wedge p* \geq v_i} v_i - v_j = \sum_{i=1}^{|MB|} mb_i - ms_i \tag{3.9}$$

by noting that we must choose $p*$ so that the induced transactions include only those agents with valuations in the match sets MB and MS.

In order to include all MB we must constrain $p*$:

$$p* \geq \min(MB) \tag{3.10}$$

and in order to include all MS we must constrain $p*$:

$$p* \leq \max(MS) \tag{3.11}$$

The above inequalities are necessary conditions for achieving TP, however we must also take care to exclude agents with valuations not in the match sets. Let MB' and MS' denote the unmatched buyer valuations and unmatched seller valuations respectively:

$$\begin{aligned} MB' &= VB - MB \\ MS' &= VS - MS \end{aligned}$$

In order to exclude valuations from these sets we must also ensure that

$$\min(MS') < p* < \max(MB') \tag{3.12}$$

Inequalities 3.10, 3.11 and 3.12 can be solved by choosing

$$p* \in [eq_a, eq_b] \tag{3.13}$$

where

$$eq_a = \max(\max(MS), \max(MB')) \tag{3.14}$$
$$eq_b = \min(\min(MS'), \min(MB)) \tag{3.15}$$

Thus yielding $S(p*) = TP$.

The solution $p*$ is known as the *equilibrium price*. Although in the general case there are a range of possible solutions, by convention when we refer to the equilibrium price we arbitrarily take a value from the middle of this range; that is:

$$p* = \frac{eq_b - eq_a}{2} \tag{3.16}$$

3.1.3 The role of the auctioneer

We have shown that we can induce individually-rational trades that result in efficient allocations provided that we know each trader's valuation v_i. However, in most practical scenarios this information is private and unobservable[1]. In a typical auction, this information is elicited through means of a bidding process, in which traders send signals[2] about their valuation to a trusted third-party called an auctioneer. The job of the auctioneer is to compute the optimal transaction set given the reported valuations. The challenge facing the auctioneer is that these signals cannot necessarily be relied upon to be truthful and accurate. Indeed, since the auctioneer allocates resource to those

[1] In game-theoretic terms valuations are part of each trader's *type* information.

[2] The term "signal" in this context derives from the theory of *signaling games* [134]. Although strictly speaking an auction is not a signaling game, the two are very strongly related. As Dutta points out [39, p. 395], in a signaling game the agents move first and then the institution responds, whereas in a mechanism design scenario the institution offers a set of moves to agents who then respond. Thus although auctions are not strictly signaling games, it can still be intuitive to think in terms of signals; by forcing agents to back up their value claims with hard cash the mechanism designer can encourage *honest signaling*. Interestingly, signaling games have also been studied in evolutionary biology in the context of the *handicap principle* [162, 18]. In the scenario under discussion, bids — that is, signals of valuation backed up with hard cash — can be thought of as "handicaps" which lead to honest signaling.

agents with higher perceived valuations, traders may have incentives to misreport their valuation.

Consider a scenario in which we have a single seller s with a valuation vs which is known to the auctioneer, and several buyers. The seller offers a single unit of resource for sale. The auctioneer elicits *reported* valuations, or *bids*, from each buyer, $\hat{VB} = \{\hat{vb}_1, \hat{vb}_2, \dots\}$, ordered such that:

$$\hat{vb}_1 \geq \hat{vb}_2 \geq \hat{vb}_3 \dots$$

which may differ from the corresponding actual valuations $VB = \{vb_1, vb_2, \dots\}$ of each buyer $B = \{b_1, b_2, \dots\}$, where b_1 is the buyer with the highest bid $\hat{vb}_1$, whose true valuation is vb_1. The reported valuations $\hat{VB}$ are known only to the auctioneer, whereas each individual buyer b_i knows only its own valuation vb_i, and bid $\hat{vb}_i$. Such a scenario is known as a single-sided sealed-bid auction, and the value vs is known as the reservation price.

The role of the auctioneer is to choose a transaction $C = \{(s, b_i, p)\}$ that maximises social welfare as defined by equation 3.7. A naive solution to this problem is to assume that agents will report their valuations truthfully; that is, $\hat{vb}_i = vb_i \; \forall i$. Accordingly, provided $\hat{vb}_1 \geq vs$:

$$\begin{aligned} MS &= \{vs\} \\ MS' &= \{\} \\ MB &= \{\hat{vb}_1\} \\ MB' &= \{\hat{vb}_2, \hat{vb}_3, \dots\} \end{aligned}$$

and

$$\begin{aligned} eq_a = \max(\max(MS), \max(MB')) = \max(vs, \hat{vb}_2) &= \hat{vb}_2 \\ eq_b = \min(\min(MS'), \min(MB)) = \min(MB) &= \hat{vb}_1 \end{aligned}$$

Thus according to equation 3.13, we should award the unit of resource to the buyer with the highest bid (the "winner"), and charge them a price $p* \in [\hat{vb}_1, \hat{vb}_2]$ anywhere between the highest bid and the 2nd highest bid. But consider the winner's ex-post[3] incentives to *misreport* their valuation for different values of $p*$ in this range. Let $U_i(x)$ denote the utility gained by trader i if it reports valuation x. In our present scenario $U_i(x) = |x - p*|$.

If we set $p* = \hat{vb}_1$, then since the utility of the winning agent is $vb_i - p*$, the winner gains $vb_1 - \hat{vb}_1$, and ex-post the winning buyer will regret having not bid a lower price w such that $\hat{vb}_2 < w < \hat{vb}_1$, since if it bids truthfully its utility will be $vb_i - vb_i = 0$,

[3]Meaning "after the fact". In economics ex-post payoffs are those that are computed once any uncertainties surrounding the payoff have been resolved, whereas ex-ante payoffs are computed under uncertainty. In the scenario under discussion the valuations of other agents are unobservable to the agent under consideration, hence until we apply the concept of Bayesian-Nash equilibrium the payoff to our agent is unknown before they choose their bid price.

whereas if it had bid w it would have received $vb_i - w > 0$. If, on the other hand, we set $p* = \hat{vb}_2$, then the buyer has no ex-post incentive to deviate from truthful bidding, since the utility of the winner is always $vb_1 - \hat{vb}_2$, regardless of the winner's reported valuation $\hat{vb}_1$.

In fact, one can show that if we set $p* = \hat{vb}_2$, then there are no *ex-ante* incentives to deviate from truthful bidding [80]. That is:

$$U_i(vb_i) \geq U_i(x) \ \forall x \forall vb_i \forall i \tag{3.17}$$

This type of auction is known as 2nd-price auction, or Vickrey auction, and the above property is known as *incentive compatibility.* [4]

3.1.4 Mechanism design

The art of designing the rules of an auction in order to bring about certain design objectives when agents act to maximise their own utility is called *mechanism design* [69], and the underlying theory is *auction theory* [80]. In a mechanism design problem, we can easily determine whether or not our design objectives are achieved provided that we know exactly how the individual traders in our mechanism will signal. However, since the behaviour of these traders is not prescribed in advance, and since they have many possible signals from which to choose, this is not a trivial problem to solve. In a mechanism design problem, we assume that individual traders will choose a signal that maximises their utility. However, this decision problem is highly complex, since, in the general case, the outcome from choosing a particular signal depends on the *joint* set of signals submitted by all agents. The theory of optimal decision-making when outcomes are the result of joint-actions is *game theory* [103]. By *solving* the game corresponding to our auction, we can, at least in theory, predict how utility-maximising traders will behave under our proposed mechanism and evaluate whether or not our design objectives are achieved.

The principle design objectives considered in auction theory are:

- *Incentive compatibility*; as defined by 3.17
- *Efficiency*; as defined by 3.6
- *Budget balance*; the mechanism can operate without external cash transfers. More formally, the full set of cash transactions C generated by the mechanism should satisfy the following constraint:

$$\sum_{c \in C : c_i \in S} c_p - \sum_{c \in C : c_i \in B} c_p = 0 \tag{3.18}$$

For single-sided mechanisms involving a single seller, auction theory demonstrates that all three of these design objectives can be achieved under wide range of conditions.

[4] Note that in order to maintain incentive compatibility, agents' bids must be binding; that is: when an agent sends a bid to an auctioneer it is committed to the possibility of paying a sum up to its bid amount — agents cannot renege on their bids.

However, the impossibility result of [94] demonstrates that no double-sided auction mechanism can simultaneously and robustly achieve all three desiderata. Thus real-world exchanges make various trade-offs between different design objectives. This is a theme we shall revisit throughout the thesis.

In the following section, we review several different double-sided auction mechanisms and briefly discuss their design properties. A fuller analysis of the design properties of these mechanisms will be conducted in Chapter 8.

3.2 The auction model

In this section I will give a formal description of different variants of the double-auction. This model is adapted from [159], [50], [28], and [98], 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 purpose of this section is to give and clear and unambiguous specification for the different auction mechanisms that we will discuss throughout the thesis. However, since the emphasis of this thesis is on empirical rather than formal methods, for brevity and conciseness I omit frame axioms from the formalism. In the following sections, if a statement cannot be proven from the axioms we shall assume that it is false.

As a final disclaimer, the model presented in this chapter does not cover multi-unit trading rules; that is, scenarios where buyers or sellers submit offers to purchase or sell more than one unit of resource at any given time. However, the formalism is easily extended to cover these scenarios as discussed in [159]

3.2.1 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*.

3.2.2 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

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

3.2.3 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:

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

3.2.4 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:

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

Thus, we have:

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

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 $x_t \in E_t$.

3.2.5 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.

3.2.6 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. These are analogous to the sets MS and MB defined in Section 3.1.2.

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

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

Similarly, for matched asks we have:

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

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

$$\forall i \;\; \text{price}(\hat{mb}_{(t,i)}) \geq \text{price}(\hat{ms}_{(t,i)}) \tag{3.19}$$

$$|\hat{MS}_t| = |\hat{MB}_t| \tag{3.20}$$

Analogous to MS' and MB', 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:

$$\begin{aligned} \rho_c = bid \wedge (\exists a \in \hat{MS'}_t : \rho_p \geq a_p) &\implies \\ \hat{MS}_{t+1} &= \hat{MS}_t \cup \{a\} \\ \wedge\, \hat{MS'}_{t+1} &= \hat{MS'}_t - \{a\} \\ \wedge\, \hat{MB}_{t+1} &= \hat{MB}_t \cup \{\rho\} \end{aligned} \tag{3.21}$$

$$\rho_c = bid \wedge (\nexists a \in \hat{MS'}_t : \rho_p \geq a_p) \implies \hat{MB'}_{t+1} = \hat{MB'}_t \cup \{\rho\} \tag{3.22}$$

$$\rho_c = ask \wedge (\exists b \in \hat{MB'}_t : b_p \geq \rho_p) \implies \hat{MB}_{t+1} = \hat{MB}_t \cup \{b\} \wedge \hat{MB'}_{t+1} = \hat{MB'}_t - \{b\} \wedge \hat{MB}_{t+1} = \hat{MB}_t \cup \{\rho\} \tag{3.23}$$

$$\rho_c = ask \wedge (\nexists b \in \hat{MB}'_t : b_p \geq \rho_p) \implies \hat{MS'}_{t+1} = \hat{MB'}_t \cup \{\rho\} \tag{3.24}$$

$$\rho_c \neq \emptyset \implies sp \in E_{t+1} \tag{3.25}$$

3.2.7 Quotes

Analogous to definitions 3.15 and 3.14, we have:

$$\hat{eq}_a(t) = \min(\min(\hat{MS'}_t), \min(\hat{MB}_t)) \tag{3.26}$$

$$\hat{eq}_b(t) = \max(\max(\hat{MS}_t), \max(\hat{MB'}_t)) \tag{3.27}$$

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. If all traders bid truthfully, then we have $\hat{eq}_a = eq_a$ and $\hat{eq}_b = eq_b$. Thus the market quote encapsulates the hypothesised range of equilibrium prices assuming truthful bidding.

3.2.8 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:

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

3.2.9 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, and these are formalized below. For now, we simply define the *clearing operation*, in which the auctioneer takes the matched shouts producing a transaction set, enforces the corresponding trades, and resets the auction state.

$$
\begin{aligned}
clr_t \in E_t \implies & \\
\Gamma_{t+1} &= \text{pay}(C_t, \Gamma_t) \\
\wedge\, \Omega_{t+1} &= \text{trans}(R_t, \Omega_t) \\
\wedge\, \hat{MS}_{t+1} &= \{\} \\
\wedge\, \hat{MB}_{t+t} &= \{\} \\
\neg clr_t \in E_t \implies & \\
C_t &= \{\} \\
\wedge\, R_t &= \{\}
\end{aligned}
$$

3.3 The clearing-house double auction

In a clearing-house (CH) double-auction, the clearing operation takes place at the end of each round:

$$
\begin{aligned}
eor_t \in E_t \implies & \\
clr_{t+1} \in E_{t+1} &
\end{aligned}
$$

The auction designer can choose from amongst several different *pricing policies* which determine exactly how the clearing operation occurs. These are formalized below.

3.3.1 Uniform pricing

A uniform pricing policy specifies that all traders with matched reported valuations (that is, all the potentially efficient trades) should all trade with each other at the reported equilibrium price (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). This variant of the CH double-auction is discussed in [50].

$$
\begin{aligned}
clr_t \in E_t \implies & \\
C_t &= \{c_1, c_2, \ldots\}
\end{aligned}
$$

where:

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

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. The design implications of different values for this constant are discussed below.

3.3.2 Discriminatory pricing

A discriminatory pricing policy, on the other hand, specifies that each pair of matched traders pays a price that is solely a function of their respective bid and ask prices. Thus, at any given time, different traders are transacting at different prices for the same commodity. This variant of the CH double-auction is discussed in [98].

$$clr_t \in E_t \implies C_t = \{c_1, c_2, \ldots\}$$

where:

$$\forall_{i \leq |\hat{MB}|}\ c_i = (\text{agent}(\hat{mb}_i), \text{agent}(\hat{ms}_i), p_i) \tag{3.28}$$

and:

$$p_i = \text{price}(\hat{mb}_{(t,i)})k + \text{price}(\hat{ms}_{(t,i)})(1-k) \tag{3.29}$$

where $k \in [0,1]$ is a constant chosen by the market designer.

3.3.3 In-order discriminatory pricing

This pricing policy specifies that trades occur at the price of the earliest submitted offer, regardless of whether it is a bid or an ask:

$$clr_t \in E_t \implies C_t = \{c_1, c_2, \ldots\}$$

where:

$$\forall_{i \leq |\hat{MB}|:\text{time}(\hat{mb}_{(t,i)}) < \text{time}(\hat{ms}_{(t,i)})}\ c_i = (\text{agent}(\hat{mb}_{(t,i)}), \text{agent}(\hat{ms}_{(t,i)}), \text{price}(\hat{mb}_{(t,i)}))$$

$$\forall_{i \leq |\hat{MB}|:\text{time}(\hat{ms}_{(t,i)}) \leq \text{time}(\hat{ms}_{(t,i)})}\ c_i = (\text{agent}(\hat{mb}_{(t,i)}), \text{agent}(\hat{ms}_{(t,i)}), \text{price}(\hat{ms}_{(t,i)}))$$

3.3.4 Properties

It is easy to see that a CH with uniform pricing is efficient provided that traders' shouts are truthful, since we will have

$$\begin{aligned} \hat{eq}_a &= eq_a \\ \hat{eq}_b &= eq_b \end{aligned}$$

and thus all transactions will occur at an equilibrium price for any $k \in [0, 1]$. However, the CH is *not* incentive compatible, and thus, in the general case, we cannot rely on utility-maximising traders to place truthful shouts. However, as [159, 154] demonstrate, there are interesting special-case exceptions when we consider extreme values of k in an auction for a single unit of commodity. When $k = 1$, we have incentive-compatibility for sellers only, but not for buyers, and when $k = 0$ we have incentive-compatibility for buyers, but not for sellers.

3.4 The continuous double-auction

In a continuous double-auction (CDA), the clearing operation is performed continuously as new shouts arrive:

$$sp_t \in E_t \implies clr_{t+1} \in E_{t+1}$$

C_t is computed as for a CH with either variant of discriminatory-pricing (Sections 3.3.2 and 3.3.3). Cliff [28] discusses a trading strategy for a CDA with in-order discriminatory-pricing.

Properties

The CDA is particularly unusual from the perspective of auction-theory, since not only is truth-telling not dominant in this institution, but allocations are *likely to be inefficient* if all agents shout truthfully. This is because the clearing operation is performed before the auctioneer has a full picture of the supply and demand in the market-place. Because clearing occurs as new shouts arrive, when the transaction set is computed from equation 3.28 there is no guarantee that the the match sets $\hat{MS}$ and $\hat{MB}$ will contain shouts corresponding to the potentially efficient trades defined by MS and MB. Indeed, there is every possibility that $\hat{MS}$ or $\hat{MB}$ will contain shouts corresponding to the potentially inefficient valuations defined by MS' and MB' since the rules in Section 3.2.6 rely on competing bids from *all* agents to arrive in order to relegate inefficient (outbid) shouts to $\hat{MS'}$ and $\hat{MB'}$.

Claim 3.2 The CDA is not always efficient when agents shout truthfully.

Proof. We will demonstrate this claim by constructing an example of a non-efficient outcome under a continuous clearing rule.

Consider a simple scenario in which we have three agents $A = \{a_1, a_2, a_3\}$ two of which are buyers $B = \{a_1, a_2\}$ and one of which is a seller $S = \{a_3\}$ with valuations:

$$\begin{aligned} V &= \{v_1, v_2, v_3\} = \{3, 2, 1\} \\ VB &= \{3, 2\} \\ VS &= \{1\} \end{aligned}$$

In order to maximise social welfare we should pair the seller a_3 together with the buyer with the highest valuation a_1, since:

$$\begin{aligned} MS &= \{1\} \\ MB &= \{3\} \\ MS' &= \{\} \\ MB' &= \{2\} \end{aligned}$$

thus we have a total possible gain from trade of $TP = v_3 - v_1 = 3 - 1 = 2$.

If these agents participate in a CDA, we see that if agents shout truthfully there is a potential to match inefficiently. Suppose that the seller a_3 is chosen to shout at $t = 0$ so that $\tau_0 = a_3$ and places a truthful shout $\rho_0 = (ask, a_3, v_3, \dots) = (ask, a_3, 1)$, so that:

$$\begin{aligned} \hat{MS}_1 &= \{\} \\ \hat{MB}_1 &= \{\} \\ \hat{MS'}_1 &= \{(ask, a_3, 1)\} \\ \hat{MB'}_1 &= \{\} \end{aligned}$$

This results in a clearing operation at $t = 1$; however, since there are no matching shouts in $\hat{MB}_1$ $\hat{MS}_1$, no transactions occur.

At the next time slice, buyer a_2 is randomly chosen to place a shout: $\tau_1 = a_2$, and shouts truthfully with $\rho_1 = (bid, a_2, v_2) = (bid, a_2, 2)$. Following the rules in Section 3.2.6, the auction state now contains:

$$\begin{aligned} \hat{MS}_2 &= \{(ask, a_3, 1)\} \\ \hat{MB}_2 &= \{(bid, a_2, 2)\} \\ \hat{MS'}_2 &= \{\} \\ \hat{MB'}_2 &= \{\} \end{aligned}$$

and since we now perform the clearing operation immediately, we will match buyer a_2 with seller a_3 yielding a total surplus of $v_2 - v_3 = 1$, and our efficiency will be only $EA = \frac{1}{TP} = \frac{1}{2} < 1$.

□

If, on the other hand, we had run this scenario using a CH instead of a CDA, the auctioneer would have waited until all agents had had an opportunity to place shouts at $t = 3$ before clearing the market, giving agent a_1 the opportunity to outbid a_2 with a shout

$$\rho_2 = (bid, a_1, v_1) = (bid, a_1, 3)$$

yielding the auction state:

$$\begin{aligned} \hat{MS}_3 &= \{(ask, a_3, 1)\} \\ \hat{MB}_3 &= \{(bid, a_1, 3)\} \\ \hat{MS'}_3 &= \{(bid, a_2, 2)\} \\ \hat{MB'}_3 &= \{\} \end{aligned}$$

which is equivalent to the optimal match sets MB and MS.

Although the CDA is potentially very inefficient under homogeneous truthful bidding, consider what happens if: (i) all agents with valuations in the match sets MB and MS place shouts at a true equilibrium price $p* \in [eq_a, eq_b]$, and (ii) all other agents (with valuations in MB' and MS') shout truthfully.

Claim 3.3 In a CDA, if all agents with valuations in MB and MS place shouts at price $p*$ and all other agents shout truthfully, we will always obtain an efficient outcome $EA = 1$.

Proof. All agents that place shouts at the same price $p*$ will eventually have shouts in the match sets $\hat{MS}$ and $\hat{MB}$ since the condition for promoting bids into the match set (equation 3.21):

$$p_c = bid \wedge \exists a \in \hat{MS'}_t : \rho_p \geq a_p$$

will always hold provided $\hat{MS'}_t \neq \{\}$ as $\rho_p = a_p = p*$. If, on the other hand, $\hat{MS'}_t = \{\}$, then by equation 3.22, $\hat{MS'}_{t+1}$ will still contain a bid with price $p*$. Similar reasoning applies to the ask promotion rules (equations 3.23 and 3.24).

By definition, those agents with valuations in MB' who shout truthfully will place shouts at lower than the equilibrium price $p*$ since

$$eq_a \leq p* \leq eq_b$$

thus from equation 3.15:

$$p* \geq max(MB')$$

Therefore their truthful bids $\rho_p = v_i$ will fail the condition $\rho_p \geq a_p$ since $a_p = p*$ and we have just shown that for these buyers $\rho_p < p*$. A similar arguments applies to sellers. Therefore our match sets $\hat{MB'}$ and $\hat{MS'}$ will contain only those shouts from traders with potentially efficient valuations in MS and MB, and since all trades will occur at the same price $p*$ regardless of which particular auction pricing rule is used, we can be sure of achieving $EA = 1$ by the reasoning in Section 3.1.2.

□

The problem with such a hypothetical trading strategy is of course that agents have no *apriori* knowledge of the true equilibrium price range. Nevertheless we have a glimpse of how high-efficiency outcomes might be achieved in a CDA *in principle.* In the following chapters, we will see that remarkably there are situations in which trading strategies can *in practice* discover the true equilibrium price range in a CDA without this knowledge of the true equilibrium price being explicitly provided by the auctioneer.

3.5 Summary and Contribution

In this chapter I have defined a space of *mechanisms.* I have drawn on previous work, and the formalism presented here to explore the design properties of various mechanisms within this space using analytical methods. However, the complexity of the mechanisms within this space is such that analytical methods on their own are unable to yield clear-cut results from the perspective of traditional auction theory; for example, none of the mechanisms presented here are *incentive-compatible* in the general case. Therefore, in the remainder of this thesis we will use empirical methods (simulation) in tandem with analytical methods in order to search the mechanism design space *heuristically.* In Chapter 5 we will return to the design space from a computational perspective, and see how auction mechanisms can be implemented and described in software, thus allowing them to be *simulated.*

Chapter 4

Trading Strategies

In the previous chapter we introduced a framework for specifying how the market allocates goods and sets prices — the rules of the market place, or the market *institution*. In this chapter, we turn our attention to the agents populating this environment. In particular, we discuss the different trading *strategies* that will be used in our models of traders' decision-making.

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 3.2.3) . Thus:

$$Z(i,t) = (bid, a_i, \zeta(i,t), 1, t) \iff a_i \in B \wedge a_i \in K_t \quad (4.1)$$

$$Z(i,t) = (ask, a_i, \zeta(i,t), 1, t) \iff a_i \in S \wedge a_i \in K_t \quad (4.2)$$

$$Z(i,t) = (\emptyset, a_i, 0, 0, t) \iff a_i \notin K_t \quad (4.3)$$

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

I will now review several classes of strategy that are commonly used in ACE research. In the following section I will discuss *non-adaptive* strategies that do not adjust their behaviour in response to changing market conditions. In Section 4.2, I will review several strategies that adapt their behaviour based on market information. Finally, in Section 4.2.4 I will discuss strategies that adjust their behaviour based solely on local feedback.

4.1 Non-Adaptive Strategies

4.1.1 The Truth-Telling Strategy

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

$$\zeta(i,t) = v_i \tag{4.4}$$

Although it is extremely simple, the truth-telling strategy is of fundamental importance, since in an incentive-compatible mechanism this strategy is guaranteed to obtain the optimal payoff for agent a_i no matter what strategies are adopted by the other agents. Of course, most double-auction mechanisms are not incentive-compatibile and hence TT is not dominant; but it is interesting to note that in a CH auction an homogeneous population of agents using TT will bring about high-efficiency outcomes ($EA = 1$) whereas in a CDA, TT will result in poor-efficiency outcomes. This is discussed further in Chapter 8.

4.1.2 The Equilibrium-Price Strategy

As we demonstrated in Section 3.4, if agents hypothetically know the *true* equilibrium price $p*$ they can coordinate on high efficiency outcomes in a wide variety of mechanisms regardless of their incentive-compatibility properties. This motivates the introduction of a *control* strategy that is useful in comparing realistic trading strategies. Agents using the Equilibrium-Price strategy (abbreviation EPS) bid at the true equilibrium price only if it is not unprofitable to do so:

$$a_i \in B \wedge p* \leq v_i \implies \zeta(i,t) = p* \tag{4.5}$$
$$a_i \in S \wedge p* \geq v_i \implies \zeta(i,t) = p* \tag{4.6}$$

As we have demonstrated this strategy will result in maximal efficiency ($EA = 1$) when all agents adopt it in a CDA mechanism.

4.1.3 The Pure Simple Strategy

In non-incentive-compatible mechanisms it may sometimes pay to shout non-truthfully. Consider a discriminatory-price clearing-house with $k = 1$ for equation 3.29. An agent who is a buyer in this mechanism $a_i \in B$, who submits a bid ρ which is subsequently matched stands to pay an amount exactly equal to their bid price, thus their surplus will be given by $v_i - \rho_p$, suggesting that they can potentially increase their surplus by bidding under their valuation, provided that their ρ_p is sufficiently high to make it into the match set C. A similar argument applies to sellers faced with a $k = 0$ mechanism.

This motivates the introduction of our first non-truthful strategy, the Pure Simple (abbreviation PS). The PS strategy bids a fixed amount above/below the agent's valuation for sellers/buyers respectively:

$$\zeta(i,t) = v_i - \mu_{it} \iff a_i \in B \tag{4.7}$$
$$\zeta(i,t) = v_i + \mu_{it} \iff a_i \in S \tag{4.8}$$

where $\mu_{it} = PS_{E_i} \in \mathbb{R}$ is a configurable parameter. Of course, the major problem we face is how to choose PS_{E_i}. On the one hand, smaller values of PS_{E_i} increase

the probability of the shout being accepted, but on the other hand this in turn may decrease the agent's surplus. Optimizing the expected surplus is non-trivial since in the general case the optimal value will depend on the mechanism that the agent is trading in (for example, if we are a seller in a $k = 1$ clearing-house we should choose $PS_{E_i} = 0$) and in non-incentive-compatible mechanisms the choice will depend on the strategies adopted by other agents, which may change over time, as well as the details of the mechanism. Thus we see that the PS strategy is very brittle. Nevertheless it is instructive to study, since the design of many other strategies in the double-auction market can be thought of as progressively more sophisticated techniques for tuning μ_{it} in response to changing market conditions.

4.1.4 The Zero Intelligence Constrained Strategy

We have seen that a very simple strategy — the TT strategy – is able to yield highly efficient outcomes ($EA = 1$) in clearing-house mechanisms, but fares poorly in continuous-clearing mechanisms. Indeed, from the perspective of the auctioneer, it is difficult to see how the market can be cleared with full efficiency in a continuous double-auction, since the auctioneer only has a partial view of the full set of potential signals representing the supply and demand in the market-place when it comes to setting prices and enforcing trades. The match sets $\hat{MS}_t$ and $\hat{MB}_t$ will contain shouts from relatively few traders, as compared with a clearing-house mechanism where the auctioneer waits until it has shouts from all traders before attempting to clear the market; using continuous clearing, the auctioneer has only a partial picture of supply and demand and cannot compute the equilibrium-price accurately.

The Zero Intelligence Constrained (abbreviation ZIC) is a slightly more sophisticated version of PS that shouts *randomly* below/above the agent's valuation:

$$\zeta(i,t) = v_i - \mu_{it} \iff a_i \in B \tag{4.9}$$

$$\zeta(i,t) = v_i + \mu_{it} \iff a_i \in S \tag{4.10}$$

where $\mu_{it} \in [0, ZIC_{E_i}] \subset \mathbb{N}$ is a discrete random variable distributed $U(0, ZIC_{E_i})$.

Gode and Sunder [58] demonstrated that this very simple strategy was able to achieve a fairly high allocative efficiency in a CDA marketplace. As Cliff comments:

> *"... the ZI-C traders scored over 99% in three experiments, and over 97% in the other two: the average efficiency for the humans was 97.9%, while for the ZI-C's it was 98.7%... thus, the main message of Gode and Sunder's paper is that allocative efficiency appears to be almost entirely a product of market structure."* [28] Page 32

4.2 Adaptive Strategies

4.2.1 The Zero-Intelligence Plus Strategy

The Zero-Intelligence Plus strategy (abbreviation ZIP) was designed as the simplest[1] possible trading algorithm that was able to yield fairly high efficiency outcomes ($EA \simeq 0.98$), as well as being able to replicate the bidding behaviour of human traders in double-auctions with continuous clearing [28]. Cliff observed that Gode and Sunder's original results [58] were not precisely replicated when agents' valuations were randomly drawn from probability distributions different to those of the original paper. Although similar allocative efficiency was observed, the distribution of transaction prices was not always as closely clustered around the equilibrium price $p*$, suggesting that a different mechanism was required to more precisely fit the data recorded from human subjects.

Each agent maintains an output-level $ZIP_\Omega(i,t)$ which determines the margin over and above their valuation that they will bid at:

$$\zeta(a_i,t) = v_i[1 + ZIP_\Omega(i,t)] \tag{4.11}$$

The output level is adjusted incrementally over time towards a target margin $ZIP_{\Omega'}(i,t)$:

$$ZIP_\Omega(i,t+1) = ZIP_\Omega(i,t) + ZIP_\gamma(i,t) \tag{4.12}$$

$$ZIP_\gamma(i,t+1) = ZIP_\gamma(i,t) \times ZIP_{\mu_i} + ZIP_\Delta(i,t) \times [1 - ZIP_{\mu_i}] \tag{4.13}$$

$$ZIP_\Delta(i,t+1) = ZIP_{\lambda_i}[ZIP_{\Omega'}(i,t+1) - ZIP_\Omega(i,t)] \tag{4.14}$$

where ZIP_{λ_i}, the learning-rate, is a constant which determines the speed of convergence, and ZIP_{μ_i}, the momentum, is a constant for dampening oscillations.

The target margin $ZIP_{\Omega'}(i,t)$ is set by observing the most recent shout placed in the market:

$$\rho : \text{time}(\rho) = t - 1$$

For sellers, if this shout resulted in a transaction $c \in C_t$, and the agent is currently trading below the observed transaction price ($\zeta(i,t) \leq c_p$), then the agent raises its target margin so that its shout price will be a small threshold, ZIP_τ, above the observed transaction price.

$$\begin{aligned} &\exists c : (c \in C_t \wedge (c_\beta = \rho \vee c_\alpha = \rho)) \implies \\ ZIP_{\Omega'}(i,t+1) &= \frac{c_p + ZIP_\tau(c_p,i) - v_i}{v_i} \; \forall_{i:(a_i \in S \wedge \zeta(i,t) \leq c_p)} \end{aligned} \tag{4.15}$$

[1] In the sense of possessing minimal state information

where the threshold function is given by:

$$ZIP_\tau(p,i) = \delta_{(1,t,i)}p + \delta_{(0,t,i)} \quad (4.16)$$

$$\delta_{(1,t,i)} \sim U(0, ZIP_{\sigma_i}) \quad (4.17)$$

$$\delta_{(0,t,i)} \sim U(0, ZIP_{\alpha_i}) \quad (4.18)$$

If the agent is currently trading above the observed price, then provided that the agent is still actively trading, the agent adjusts its price towards a small threshold below the observed transaction price:

$$\exists c : (c \in C_t \wedge (c_\beta = \rho \vee c_\alpha = \rho)) \implies$$
$$ZIP_{\Omega'}(i,t+1) = \frac{c_p - ZIP_\tau(c_p,i) - v_i}{v_i} \forall_{i:(a_i \in S \wedge \zeta(i,t) > c_p)} \quad (4.19)$$

If the last shout did not result in a transaction then active agents will adjust their prices towards a small threshold below the shout price regardless of their current price:

$$\nexists c : (c \in C_t \wedge (c_\beta = \rho \vee c_\alpha = \rho)) \implies$$
$$ZIP_{\Omega'}(i,t+1) = \frac{\rho_p - ZIP_\tau(\rho_p,i) - v_i}{v_i} \forall_{i:(a_i \in S)} \quad (4.20)$$

Correspondingly, for buyers:

$$\exists c : (c \in C_t \wedge (c_\beta = \rho \vee c_\alpha = \rho)) \implies$$
$$ZIP_{\Omega'}(i,t+1) = \frac{c_p - ZIP_\tau(c_p,i) - v_i}{v_i} \forall_{i:(a_i \in B \wedge \zeta(i,t) \geq c_p)} \quad (4.21)$$

$$\exists c : (c \in C_t \wedge (c_\beta = \rho \vee c_\alpha = \rho)) \implies$$
$$ZIP_{\Omega'}(i,t+1) = \frac{c_p + ZIP_\tau(c_p,i) - v_i}{v_i} \forall_{i:(a_i \in B \wedge \zeta(i,t) < c_p)} \quad (4.22)$$

$$\nexists c : (c \in C_t \wedge (c_\beta = \rho \vee c_\alpha = \rho)) \implies$$
$$ZIP_{\Omega'}(i,t+1) = \frac{\rho_p + ZIP_\tau(\rho_p,i) - v_i}{v_i} \forall_{i:(a_i \in B)} \quad (4.23)$$

4.2.2 Kaplan's Sniping Strategy

Todd Kaplan's sniping strategy (abbreviation TK) waits until the last moment before attempting to "steal the bid"; sniping agents remain inactive in the background until the state of the auction is in their favour or time is running out, at which point they place truthful shouts [51].

Parameter name	*Semantics*	*Range*
ZIP_{σ_i}	Scaling factor	$\in \mathbb{R}$
ZIP_{α_i}	Absolute perturbation	$\in \mathbb{R}$
ZIP_{λ_i}	Learning rate	$\in \mathbb{R}$
ZIP_{μ_i}	Momentum	$\in [0, 1]$

Table 4.1: Parameters for the ZIP strategy

Variable	*Semantics*
$ZIP_{\gamma}(i,t)$	Cumulative discounted momentum
$ZIP_{\Omega}(i,t)$	The current output-level for agent i at time t
$ZIP_{\Delta}(i,t)$	The level of adjustment for agent i at time t
$ZIP_{\Omega'}(i,t)$	The target margin for agent i at time t

Table 4.2: State variables for the ZIP strategy

Let Y_t denote the set of transactions that occurred in the previous day's trading.

$$\{v_1, v_2, \dots\} = Y_t = \bigcup_{i:day_i = day_t - 1} C_i$$

The set is ordered on transaction price:

$$a \leq b \iff price(v_a) \leq price(v_b)$$

Let T denote the number of ticks until the next clearing operation. Let σ_t denote the *market spread*:

$$\sigma_t = |e\hat{q}_b(t) - e\hat{q}_a(t)| \tag{4.24}$$

Kaplan snipers shout truthfully:

$$\zeta(i,t) = v_i$$

but only when the market is in their favour:

$$a_i \in B \wedge e\hat{q}_a(t) < \min(Y_t)_p \implies K_{t+1} = K_t \cup \{a_i\} \tag{4.25}$$
$$a_i \in S \wedge e\hat{q}_b(t) > \max(Y_t)_p \implies K_{t+1} = K_t \cup \{a_i\} \tag{4.26}$$
$$a_i \in B \wedge \frac{\sigma_t}{e\hat{q}_a(t)} < KAP_{\sigma_i} \implies K_{t+1} = K_t \cup \{a_i\} \tag{4.27}$$
$$a_i \in S \wedge \frac{\sigma_t}{e\hat{q}_b(t)} < KAP_{\sigma_i} \implies K_{t+1} = K_t \cup \{a_i\} \tag{4.28}$$

or time is running out:

$$T < KAP_{\tau_i} \implies K_{t+1} = K_t \cup \{a_i\} \tag{4.29}$$

Parameter name	*Semantics*
KAP_{τ_i}	The time factor
KAP_{σ_i}	The spread factor

Table 4.3: Kaplan parameters

4.2.3 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 [57].

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:

$$\hat{HS}_0 = \hat{HB}_0 = \hat{HS'}_0 = \hat{HB'}_0 = \{\} \tag{4.30}$$

As shouts are *placed* (Section 3.2.6) they are recorded in the history of *unaccepted* shouts:

$$\hat{HS'}_{t+1} = \hat{HS'}_t \cup \rho \iff \rho \in \hat{MS'}_t \tag{4.31}$$

$$\hat{HB'}_{t+1} = \hat{HB'}_t \cup \rho \iff \rho \in \hat{MB'}_t \tag{4.32}$$

As shouts are *matched* (Section 3.2.6) they are recorded in the history of *accepted* shouts:

$$\hat{HS}_{t+1} = \hat{HS}_t \cup \rho \iff \rho \in \hat{MS}_t \tag{4.33}$$

$$\hat{HB}_{t+1} = \hat{HB}_t \cup \rho \iff \rho \in \hat{MB}_t \tag{4.34}$$

Note that the history is unaffected by the clearing operation (Section 3.2.9), hence once a shout is recorded as accepted it remains so, unless it is removed due to memory-size restrictions as defined below.

Let

$$\vec{hs_t} = \{hs_{(1,t)}, hs_{(2,t)}, \ldots, hs_{(GD_N,t)}\} \tag{4.35}$$

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

$$\vec{hb_t} = \{hb_{(1,t)}, hb_{(2,t)}, \ldots, hb_{(GD_N,t)}\} \tag{4.36}$$

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

$$clr_t \in E_t \implies h_{t+1} = h_t + |C_t| \mod GD_N \tag{4.37}$$

$$\exists \rho : \rho_t = t \wedge \rho_c = ask \implies \tag{4.38}$$

$$hs_{(h_t+1,t+1)} = hs_{(h_t+1,t)} + 1 \tag{4.39}$$

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;

$$GD_{TAG(p,t)} = |\{\rho : \rho \in \hat{HS}_t \wedge \rho_p \geq p\}| \tag{4.40}$$

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

$$GD_{BG(p,t)} = |\{\rho : \rho \in (\hat{HB}_t \cup \hat{HB}'_t) \wedge \rho_p \geq p\}| \tag{4.41}$$

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

$$GD_{RAL(p,t)} = |\{\rho : \rho \in \hat{HS}'_t \wedge \rho_p \leq p\}| \tag{4.42}$$

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

$$GD_{TBL(p,t)} = |\{\rho : \rho \in \hat{HB}_t \wedge \rho_p \leq p\}| \tag{4.43}$$

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

$$GD_{AL(p,t)} = |\{\rho : \rho \in (\hat{HS}_t \cup \hat{HS}'_t) \wedge \rho_p \leq p\}| \tag{4.44}$$

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

$$GD_{RBG(p,t)} = |\{\rho : \rho \in |\{\rho \in \hat{HB}'_t \wedge \rho_p \geq p\}| \tag{4.45}$$

Parameter name	*Semantics*
GD_N	The memory size

Table 4.4: GD parameters

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

$$GD_{\hat{pa}(p,t)} = \frac{GD_{TAG(p,t)} + GD_{BG(p,t)}}{GD_{TAG(p,t)} + GD_{BG(p,t)} + GD_{RAL(p,t)}} \tag{4.46}$$

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

$$GD_{\hat{pa}(p,t)} = \frac{GD_{TBL(p,t)} + GD_{AL(p,t)}}{GD_{TBL(p,t)} + GD_{AL(p,t)} + GD_{RBG(p,t)}} \tag{4.47}$$

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:

$$GD_{E(p,i,t)} = (v_i - p_p)GD_{pa(p)} \tag{4.48}$$

and for seller i:

$$GD_{E(p,i,t)} = (p_p - v_i)GD_{pa(p)} \tag{4.49}$$

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

$$\zeta(i,t) = \arg\max_{p*} GD_{E(p*,i,t)} \tag{4.50}$$

4.2.4 Reinforcement-learning Strategies

The adaptive strategies in the previous sections are general-purpose in the sense that they do not rely on any of the underlying implementation details of the auction mechanism, such as the particular clearing rule that is used. They do, however, rely on certain market-data being made available; the ZIP and GD strategies rely on the shouts of other agents to be made public information, and the TK strategy relies on public market-quote data. The strategies in this section, in contrast, 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:

$$\zeta(i,t) = v_i + RL_{\lambda_i}(t)RL_{\mu_i} \iff a_i \in S \tag{4.51}$$

$$\zeta(i,t) = v_i - RL_{\lambda_i}(t)RL_{\mu_i} \iff a_i \in B \tag{4.52}$$

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

$$RL_{\rho_i}(t) = \Gamma_t(a_i) - \Gamma_{t-1}(a_i) \tag{4.53}$$

The function $RL_{\lambda_i} : \mathbb{N} \rightarrow \Theta_i$ 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 4.5: Reinforcement-learning parameters

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:

$$RL_{\lambda_i} = \delta_{i_t} \tag{4.54}$$

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. Functionally it is equivalent to the ZIC strategy (Section 4.1.4).

The Roth-Erev learning algorithm

The Roth-Erev algorithm (abbreviation RE) is designed to mimic human game-playing behaviour in extensive form games [43]. Agents bid probabilistically according to:

$$RL_{\lambda_i}(t) = RE_i(t) = \delta_{i_t} \tag{4.55}$$

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

$$P(\delta_{i_t} = x) = RE_p(x, i, t) \tag{4.56}$$

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

$$RE_q(\theta, a_i, t_0) = \frac{RE_{s_i}}{RL_{k_i}} \tag{4.57}$$

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

$$\begin{aligned} RE_q(\theta, a_i, t) = (1 - RE_{\rho_i}) RE_q(\theta, a_i, t-1) \\ + RE_\epsilon(\theta, a_i) \end{aligned} \tag{4.58}$$

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

$$RE_\epsilon(\theta, a_i, t) = RL_{\rho_i}(t-1)[1 - RE_{\eta_i}] \iff \theta = RE_i(t-1) \tag{4.59}$$

$$RE_\epsilon(\theta, a_i, t) = RL_{\rho_i}(t-1)\frac{RE_{\eta_i}}{RL_{k_i}-1} \iff \theta \neq RE_i(t-1) \tag{4.60}$$

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

$$Q_{i_t} = \sum_{\theta \in \Theta_i} RE_q(\theta, a_i, t) \tag{4.61}$$

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

$$RE_p(\theta, a_i, t) = \frac{RE_q(\theta, a_i, t)}{Q_{i_t}} \tag{4.62}$$

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 4.6: Parameters for the Roth-Erev learning algorithm

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

Nicolaisen, Petrov and Tesfatsion [98] (abbreviation NPT) used a modified version of the Roth-Erev algorithm $RL_{\lambda_i}(t) = RE'_i(t)$ where $RE'_i(t)$ is computed identically to $RE_i(t)$ but for a modification to the experience function:

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 4.7: State variables for the Roth-Erev learning algorithm

$$RE_{\epsilon'}(\theta, a_i, t) = RL_{\rho_i}(t-1)[1 - RE_{\eta_i}] \iff \theta = RL_I(t-1) \quad (4.63)$$

$$RE_{\epsilon'}(\theta, a_i, t) = RE_{q_i} \frac{RE_{\eta_i}}{RL_{k_i}-1} \iff \theta \neq RE_i(t-1) \quad (4.64)$$

The use of this algorithm is discussed further in Section 6.3.

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 [147]. 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:

$$SQ_Q(\theta, a_i, t+1) = SQ_Q(\theta, a_i, t) + SQ_{\alpha_i}\left[RL_{\rho_i} + SQ_{\gamma_i} \max_{\theta'} SQ_Q(\theta', a_i, t) - SQ_Q(\theta, a_i, t)\right] \quad (4.65)$$

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:

$$RL_{\lambda_i}(t) = \delta_{it} \iff \epsilon'_{it} \leq SQ_{\epsilon_i} \quad (4.66)$$

$$RL_{\lambda_i}(t) = \arg\max_{\theta*} SQ_Q(\theta*, a_i, t) \iff \epsilon'_{it} > SQ_{\epsilon_i} \quad (4.67)$$

where $\epsilon'_{it} \in \mathbb{R}$ is a random variable distributed $U(0,1)$ and $\delta_{it} \in \mathbb{N}$ is a discrete random variable distributed $U(0, RL_{k_i} - 1)$.

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

Table 4.8: Parameters for the stateless Q-Learning algorithm

4.3 Summary and Contribution

In this section I have presented several classes of trading strategy from the double-auction literature. The main contribution of this chapter has been to formulate all of these strategies within the common framework defined in the previous chapter. In so doing, I have been able to formulate these strategies in such a way that they can be seen to be applicable in several different variants of the double-auction market. For example, although the ZIP and GD strategies were originally formulated in the context of a market with continuous-clearing, nothing in their formulation herein depends on the form of the clearing-rule that is used. Indeed, in Chapter 8, we shall use a common set of strategies from this chapter to explore the design implications of different double-auction mechanisms.

Chapter 5

Simulation Framework

As we saw in Chapter 3, many of the variants of the double-auction are extremely difficult to analyse using traditional analytical methods from classical game theory and auction theory. In such cases, *simulations* of double-auction markets have been used to shed light on some of the grey areas that are difficult to analyse using existing theories. This methodology has come be to known as Agent-based Computational Economics (ACE) [139]. In this chapter I give an overview of the simulation framework that was used to conduct the experiments that are reported on in later chapters.

As with any software engineering problem, in choosing an appropriate software framework in which to implement an ACE simulation it is important to consider the requirements that the software needs to meet. In Section 5.1, I give an overview of the typical requirements addressed by simulation software in general, and I then proceed to give an overview of some commonly-used simulation frameworks categorised according to the functionality that they provide. In Section 5.1.7 I give an overview of my specific requirements for simulating the double-auction market, and give an outline of the design of the system in terms of the formal model specified in Chapter 3.

5.1 An overview of multi-agent simulation

5.1.1 Simulating a MAS verses implementing a MAS

Software for *simulating* multi-agent systems typically addresses different requirements from that designed to *implement* multi-agent systems. Although it is natural to view a MAS implementation as its own simulation, there are a number of problems with such an approach, which I shall address in turn.

Firstly, ideally we would like the outcome of a simulation experiment to be exactly reproducible given the initial conditions of the experiment. This is not always possible in a MAS implementation since many environmental factors will be beyond the experimenter's control. For example, the precise outcome of an experiment may depend on the exact timing with which an agent responds to a particular message, and this time interval will depend on factors beyond the experimenter's control, such as the memory

and CPU currently available to the agent.

Secondly, when we come to analyse the results of a simulation, we often need to generalise beyond a single run of an experiment with a single set of initial conditions. Typically, we generalise by taking many samples of free initial variables and running the experiment many times for each sample. Simulation frameworks are equipped to log data from the outcome of each experiment to a format suitable for analysis using statistical analysis software, such as MAT-LAB.

Thirdly, the performance considerations of a simulation are qualitatively different to that of an implementation. The software architecture of a MAS implementation is driven by real-world requirements that do not always hold in a simulation context, and once these requirements are relaxed, alternative architectures can yield significant performance improvements. For example, trading agents in the real-world need to be able to run on different machines due to commercial and practical considerations. This distributed parallelism is detrimental to raw system-level performance however, since inter-host network communication overheads dominate other performance considerations. By running all agents on the same host we can achieve several orders of magnitude performance increase since inter-agent communication can be achieved with the negligible cost of an intra-process method invocation. This would be an impractical solution for a real MAS trading implementation, and would not achieve a reduction in elapsed auction times anyhow, since much of the processing involved needs to be synchronised with sporadic real-world events (such as waiting for a human to determine their valuation for an item). However, such considerations do not apply in a simulation context, and by relaxing these constraints we can achieve a significant gain in performance.

Similarly, much of the technical complexity of a real MAS implementation addresses requirements that are not present in a simulation context. MAS implementations need to be robust against system failures, and they need to respond quickly to real-time asynchronous events. This requires a highly-parallel software architecture, involving, for example, many threads of execution running simultaneously. This in turn necessitates advanced, and costly, programming techniques for dealing with the common defects, such as race conditions [22], that can arise in parallel applications. Such considerations do not apply in agent-based simulation, since real-time parallelism can be simulated using a sequential program, and this greatly reduces the complexity of the software (and hence the potential for bugs). Note that since we typically run the simulation very many times (with different samples of free variables), we can easily scale-up the performance of our experiments by running different samples on different hosts, despite the fact that we are using a sequential software architecture.

Finally, any MAS interacts at some point with a set of non-agent components, viz the environment. In an ACE scenario, for example, the environment might constitute economically relevant characteristics of the human owners of agents, such as their utility functions. Unlike the agents in a MAS implementation, the environment is not a software entity, and cannot be directly ported to an agent-based simulation. Rather, the environment itself must simulated. Agent-based simulation toolkits provide Monte-Carlo functionality for the abstract statistical simulation of environmental factors, which are often modeled as stochastic processes. In an ACE scenario, for example, rather than explicitly modelling the socio-biological formation of human preferences, we may

assume for convenience that preferences are drawn from some random distribution. Hence a key feature of any simulation toolkit is a library of pseudo-random number generators (PRNG). The PRNGs provided in simulation toolkits are more advanced than those provided in standard programmer's libraries, such as, for example, the rand() function provided by Unix's libc library, which suffer from small periods and predictable correlations between numbers in the sequence which can skew the results of experiments [102]. A good simulation toolkit will provide high quality PRNGs, such as the Mersenne Twister PRNG [85], with extremely large periods, low statistical correlation, and the ability to produce random numbers according to arbitrary (non-uniform) distributions.

In summary, when developing a system to simulate a multi-agent system, it is important to choose a framework or toolkit that is specifically designed for agent-based *simulation*, as opposed to toolkits such as JADE [7] that are designed for *implementing* multi-agent systems.

5.1.2 Different approaches to simulating time

As discussed in the previous section, for practical purposes we prefer to simulate the parallelism of events using sequential computation, rather than execute the simulation of multiple simultaneous events in parallel in real-time. This necessitates a framework for computing the outcome of events that occur simultaneously. Since computations in the simulation corresponding to these events occur at different times from the times that the events would have occurred at in a real system, this results in two distinct notions of time, viz "simulation-time" as opposed to "real-time" (also known as "wall-clock time"). Since time plays an important causal role in any model, it is important to be able to time-stamp events with the simulation-time that they occurred and provide the current simulation time to all entities in the model. Hence, we need to simulate the progression of simulation-time and its causal role. There are several approaches to simulating time in a model:

5.1.3 Continuous time models

Many physical processes are characterised by smooth and continuous changes in time-dependent variables. For example, the velocity of a projectile is a continuous function of its acceleration and time, and may vary smoothly over an arbitrarily small interval of time. Processes such as these are typically modeled using systems of differential equations. Many simulation toolkits exist for approximately solving, and analysing the dynamics of models expressed as differential or difference equations.

Differential equation models are common in analytical microeconomics. Such models are applicable approximations of real marketplaces when there are very large numbers of participants in the market since individual characteristics of the participants play a less significant role and the entities in the system can be treated as simple and homogeneous particles. However, these models break down when the number of participants becomes very small and the individual and strategic characteristics of the participants become more prominent.

Agent-based models address this issue by providing a richer structure with which to model market participants. In such models, macro-level variables describing the ensemble of agents no longer vary smoothly with time. This necessitates alternative approaches to temporal modelling.

5.1.4 Discrete-event simulation

Discrete-event simulation frameworks [5, 54] model time in discrete quanta called "ticks". Intuitively, a tick can be thought of as an "instant" of time. During the simulation of a tick – the "tick cycle" – entities (agents) in the simulation signal which agents they interact with during that instant of time by sending *events* to each other. Individual events specify the exact nature of the interaction between agents. In an auction simulation, for example, an auctioneer agent may send an end-of-auction event to all trading agents in the auction when it has closed. At the end of tick cycle, once events have been exchanged, each entity updates its internal state in response to any events it has received. Since an entity can have multiple events in its event queue, it can take into account the causal effect of multiple simultaneous events when it comes to computing its updated state.

5.1.5 Agent decision functions

In an ACE simulation agents often need to make intelligent decisions in their resource utilisation and acquisition behaviour. Modelling intelligent decision making behaviour is one of the central problems in Artificial Intelligence research, and there are as many approaches to modelling agent decision functions in agent-based simulations as there are schools of thought in AI.

The intelligent-agents community has traditionally favoured *symbolic* approaches, such as the class of Belief-Desire-Intention (BDI) models [157]. In an ACE scenario, however, the most important aspect of an agent's goals is their ordering with respect to the agent's preferences; for example, agents may act to maximise their expected *utility*. In the field of agent-based electronic commerce, this has led to the adoption of *numerical* methods based on dynamic programming [8, 36], such as (multi-agent) reinforcement learning[1], in which the symbolic concept of a goal is replaced by a numerical reward value.

Many agent-based simulation frameworks have been developed by the Artificial-Life (ALife) community. Agents in Alife models are imbued with very little intelligent behaviour at the outset; rather, intelligent behaviour emerges collectively from the complex interactions between agents equipped with relatively crude decision making machinery. Connectionist approaches such as neural-networks and evolutionary approaches such as genetic-algorithms, are popular in such models [78].

Since simulation is the main methodology used in ALife research, ALife software toolkits tend to be the most mature in terms of simulation functionality. Correspondingly, since empirical methods are relatively rare in MAS research, there are few frameworks for *simulating* BDI agents, as opposed to implementing BDI agents. Thus when

[1] See Chapter 4 of [137] for an explanation of the relationship between reinforcement-learning and dynamic programming.

conducting simulations of BDI agents, it is sometimes necessary to develop software to integrate functionality provided by BDI software toolkits with that provided by simulation toolkits.

5.1.6 Extensibility and integration

When conducting research via simulation it is often necessary to *extend* the existing functionality of the system. Although all frameworks provide the ability to configure simulations, the desired behavior cannot always be implemented by configuring the existing components provided by the framework. In this case it is necessary for the researcher to implement the desired behaviour by writing their own code. Toolkits take two main approaches to allowing extensibility:

- scripting in a custom language; and
- introducing new classes and methods via inheritance

The former is sometimes preferred when the researcher is not a skilled programmer, but in general the latter approach is much preferable. The disadvantages of a scripting approach are considerable; third-party libraries, for example: libraries providing BDI or reinforcement-learning functionality, cannot be used; the language may not necessarily well-supported or well-known by the community and if it is an interpreted language it may cause performance issues.

In judging whether or not a toolkit is extensible via object-oriented programming, one needs to ask the following questions:

- Is the source code for the original framework available?
- Is it written in an object-oriented language?
- Is it available under an open-source license?
- Is there comprehensive API documentation?
- Is the code well-structured and designed for extensibility?

Extensible software is typically characterized by very many small classes each with a clear functional role, and each with many small methods. Software designed in a monolithic fashion with a few large classes, or with very long methods, is hard to extend.

5.1.7 Requirements

As discussed, when selecting an appropriate foundation on which to build a simulation system it is important to review the requirements that software is to meet. In this section, I review the key requirements that drove the development of simulation software used for my research.

- *Large numbers of auctions with different sets of agents*

  I am interested in applying techniques from evolutionary computation to negotiation and market design problems. This involves running a particular trading scenario a large number of times with different sets of evolving agents, and/or evolving mechanisms. A typical experiment may, for example, require evaluating market outcomes over 10^4 generations of evolution and require of the order of 10^6 auctions to be run in total.

- *A variety of auction protocols*

  My interest in auctions arises from their generality; ie their applicability to a wide range of scenarios in negotiation and market design problems. In order to be general enough, our simulation software needed to support a wide variety of configurable auction protocols, including *double auctions* in which both buyers and sellers submit offers, and multi-unit auctions in which multiple units of a commodity are traded.

- *The ability to change the rules of the auction*

  Because I am interested in market design, I need to run experiments where I vary the rules of the auction. These variations in auction rules may not always be taken from the set of known analysed auction rules.

- *The ability to experiment with a wide variety of trading strategies*

  I am interested in running simulations with a wide variety of behavioural strategies.

The key requirements can be summarised as *performance* and *extensibility*. During the course of my research, I could not find any existing auction simulation software that supported the above requirements, and so I started development on the *Java Auction Simulator API* (Abbreviation JASA) project[2]. JASA is a high-performance extensible auction simulator written in Java. JASA is built on top of the Repast multi-agent simulation framework [101, 125]. In the following section I review many of the common simulation frameworks available and explain why Repast was chosen.

5.1.8 Software listing

In this section, we give a brief overview of some commonly-used general-purpose simulation frameworks that might be suitable for analysing ACE problems.

Swarm

Swarm is one of the most famous ALife software toolkits and has been continually improved by an active community of users and developers since the early 1990s [89]. It provides an API for discrete-event simulation.

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

Features
high-quality PRNGs
discrete-event simulation
spatial modelling
real-time visualisation tools
Advantages
It is well-supported and well-known by researchers
Open source
Disadvantages
It is written in Objective-C which is an obscure programming language which is not well supported, although recently a JAVA interface has been provided using JNI (Java Native Interface)
Languages and Platforms
Objective-C
Windows
Unix

MAML - Multi-Agent Modelling Language

MAML is an extension to Swarm that provides a higher-level scripting language that is simpler to use than Objective-C [62]. The goal is to allow researchers from the social sciences, who are not necessarily skilled programmers, to quickly develop simulations.

Features
high-quality PRNGs
discrete-event simulation
spatial modelling
real-time visualisation tools
Advantages
It is well-supported and well-known by researchers
Open source
Disadvantages
It is written in Objective-C which is an obscure programming language which is not well supported, although recently a JAVA interface has been provided using JNI (Java Native Interface)
Languages and Platforms
Objective-C
Windows
Unix

RePast

The RePast toolkit is inspired by Swarm, but is written entirely in Java, and the ultimate design goals of this system are more MAS-oriented than ALife-oriented [101, 125].

Functionality
high-quality pseudo-random number generation (Mersenne Twister)
discrete-event simulation
spatial modelling
real-time visualisation tools
Advantages
Open source
Extensible
Core simulation functionality is relatively mature and robust; Uses the CERN colt library for high-performance scientific computing.
Disadvantages
MAS-oriented features are relatively immature. There is currently no explicit reinforcement-learning or BDI support.
Languages and Platforms
Java
Multi-platform

Desmo-J

Desmo-J provides raw discrete-event simulation functionality[3]. It uses the standard Java PRNG, but the API should allow other PRNGs to be plugged in.

Functionality
discrete-event simulation
Advantages
Highly-flexible
well-designed API
Disadvantages
Minimal functionality is provided beyond discrete-event modelling
Languages and Platforms
Java
Multi-platform

AScape

AScape is a Java-based discrete-event simulation framework with an emphasis on spatial modelling of agents[4].

[3] `http://sourceforge.net/projects/demoj`
[4] `http://www.brookings.edu/es/dynamics/models/ascape/`

Functionality
discrete-event simulation
spatial modelling - including diffusion modelling
visualisation
Advantages
Oriented towards social-science research
Disadvantages
No high-quality PRNG algorithms provided
monte-carlo functionality somewhat ad-hoc.

deX - Dynamic Experimentation Toolkit

deX is a C++ framework for building multi-agent systems with an emphasis on three-dimensional visualisation[5].

Functionality
high-quality pseudo-random number generation
discrete-event simulation
spatial modelling
real-time visualisation tools, including 3D
Advantages
High-performance
Disadvantages
Licensing agreement unclear
Source-code hard to obtain
Languages and Platforms
C++
Linux

5.1.9 Choice of toolkit

There are a great many agent simulation toolkits available in the software domain. I have reviewed several that were popular at the time of writing. I chose the RePast simulation framework [101, 125] as the basis of my simulation software.

5.1.10 Choice of language

In order to be truly extensible, the system must give researchers the ability to program their own trading strategies, and auction mechanisms. This necessitates the use of a general purpose programming language. Rather than creating customised programming languages for writing trading strategies and auction rules, I decided to use the Java programming language. Java suits our design goals because it supports the following features:

- *Extensibility via inheritance.*

[5] `http://dextk.org/dex/index.html`

Researchers can create auction mechanisms and trading strategies from a set of reusable software *components*, whose functionality they can extend and modify using inheritance. Base classes can be provided which serve as *skeletons* for further development. For example, a base class for auction mechanisms is provided as part of the system; this class encapsulates common behaviour for all types of auction. A researcher can use this base class as a skeleton, or template, which they can extend, by for example, replacing the generic method for determining a clearing price, with custom code to implement a specific pricing policy.

- *Performance.*

  In some circles, the Java programming language has gained a reputation for performance problems. This reputation, however, derived from very early implementations of the Java Virtual Machine (JVM). Although for some benchmarks Java is not *quite* as fast as C++, modern JVMs (versions 1.4 and above) are several orders of magnitude faster than the version 1.1 JVM, and many benchmarks demonstrate superior performance for Java over C++ in some cases[6]. It is now widely acknowledged that Java is mature enough to be used for for high-performance numerical computing [11, 93]. Part of Java's previous poor reputation in this area may be due the different style of performance optimization required, as summarised by Shirazi:

  > *"There is a general perception that Java programs are slow. Part of this perception is pure assumption: many people assume that if a program is not compiled, it must be slow. Part of this perception is based in reality: many early applets and applications* were *slow, because of nonoptimal coding, initially unoptimized Java Virtual Machines (VMs), and the overhead of the language. In earlier versions of Java, you had to struggle hard and compromise a lot to make a Java application run quickly. More recently, there have been fewer reasons why an application should be slow. The VM technology and Java development tools have progressed to the point where a Java application is not particularly handicapped."* [129, p. 1]

- *Ease of use.*

  Java is relatively easy to learn, compared with, for example, C++, and it is also well-established; many researchers already possess Java programming skills.

- *Proximity to agent-oriented programming.*

  Although it is not itself an agent-oriented programming language, Java has many features in common with other Object-oriented languages that make development of MAS simulations relatively straight-forward. For example, different types of agents can be represented as classes; individual agents can be instantiated as objects and agents can communicate amongst one another by invoking methods on each other.

[6] For example, see `http://www.kano.net/javabench/`.

5.2 Engineering Methodology

As Ropella *et al.* [116] point out, building a simulation of a system as opposed to implementing a system poses unique software engineering challenges, which I shall examine in the following sections.

5.2.1 Unit testing

Functional testing is very difficult when simulating complex-adaptive systems (CAS), such as the double-auction market place. When developing a traditional software system, we typically have a set of well-defined system requirements that specify the exact macro-level behaviour of the system. These requirements are *complete*, in the sense that unanticipated behaviours which have not been specified in advance are considered undesirable[7]. However, when we are simulating a CAS, we are often interested in *emergent* behaviour that has not been specified in advance; in traditional software engineering, such behaviour is an obvious sign of a defect, whereas in CAS research, it is the entire point of the exercise. That is not to say, however, that software defects are unimportant when we come to simulate a MAS. On the contrary, we need to have as much confidence as possible that the effects we observe are a result of the actual assumptions that we *state* and not a result of an incorrect implementation of these underlying assumptions.

In traditional software engineering, we can laboriously, but methodologically, test the system to see that its behaviour matches a concrete specification. However this is problematic in CAS modelling since the "specification" corresponds to a set of stated assumptions about the domain, and does not take the form of statements about the macro-level behaviour of the system. Hence, so-called *black-box* testing methods cannot be applied to CAS modelling software.

Therefore, when developing CAS software, we need to place much more emphasis on *glass-box* testing methods, which attempt to verify the correctness of individual software components rather than system-level behaviour. The approach that I have adopted is that of automated unit-testing [68]; each class (component) in the system has a corresponding class which is responsible for testing the class under consideration by invoking each of its methods with different parameters. The testing class then verifies that each methods returns the expected result given the supplied parameters.

This approach is especially beneficial in developing research software, since it results in an automated suite of tests that can be quickly used to *regression test* the system after we have made changes to the software. In traditional software engineering, once the software has been released, code changes are prohibitively expensive, partly because of the cost of testing to ensure that changes have not resulted in new defects to other parts of the system as Brooks comments:

> *"Also as a consequence of the introduction of new bugs, program maintenance requires far more system testing per statement written than*

[7]For example, many security vulnerabilities in software typically fall into this category of defect; a vulnerability arises when an attacker is able to exploit unanticipated behaviour from a piece of a software to their own advantage.

> *any other programming. Theoretically, after each fix one must run the entire batch of test cases previously run against the system, to ensure that it has not been damaged in an obscure way. In practice, such regression testing must indeed approximate this theoretical idea, and it is very costly."* [15, p. 122]

Therefore, traditional methodologies place an emphasis on freezing the code to further changes prior to release of the software- that is, once the software is "finished". However, research software is rarely finished in this sense of the word, hence it is necessary to use engineering methods that are robust to software changes, such as for example, frequent, *automated* regression-testing [68].

Increasingly, it has been realised that this also true of commercial software, and this has led to the development of software engineering methods that are able to cope with changing requirements – so called *agile* software engineering methodologies [6], in which regular automated unit-testing is one of the principle techniques. During the development of JASA, I have incorporated various other techniques from agile engineering methodologies, including the principle of making releases of the software available to other researchers as early and as frequently as possible ("release early, release often" [114]), as discussed below.

5.2.2 Replication of existing experiments

For some problems involving the use of simulation, we have a concrete physical system corresponding to the simulation; for example, when modelling meteorological phenomena, we have a physical system – the atmosphere – which we can model using meteorological simulation software. In such scenarios, we can use the observed behaviour of the physical system as a specification for the corresponding software simulation.

In much of the ALife and Multi-Agent Systems research into the double-auction market, however, the emphasis is not so much on fine-grained quantitative or predictive models of actual real-world instances of this institution, but rather, on assessing the qualitative behaviour of such systems under different assumptions. This has resulted in a proliferation of terminology for *abstracta*: entities that exist solely in different researcher's models. Whereas meteorologists know exactly what they mean when they talk about, for example, a "cirrus cloud formation", since they can, in a sense, simply point out of the window at one, it is more problematic for ALife researchers to know exactly what is meant by, for example, a ZIP agent [28], since this is neither an entity that exists in the real-world, nor an abstraction that possesses a simple and elegant mathematical description within a well-defined theoretical framework. Indeed, the most precise and concise definitions of a ZIP agent utilise some form of pseudo-code.

This presents methodological hurdles, however, since code is rarely directly portable across different experimental frameworks. Thus researchers typically have to re-engineer such components from scratch when conducting experiments.

If one is reporting on results that are hypothesised to be contingent on a particular trading strategy such as ZIP, one has to be careful that we have an agreed-upon definition of what ZIP actually is. Definitions in the form of code, however, present

a challenge, since it is rarely possible to prove whether two pieces of code exhibit identical behaviour, even for very simple programs.

Therefore, when adopting abstracta with procedural definitions into one's experimental framework, it is crucial to attempt to determine as far as possible whether the imported entity behaves as specified. One approach that I have adopted in this thesis is to rely on *replication attempts*, in which I attempt to replicate as precisely as possible the results reported in the original work. These are discussed further in Chapter 6.

5.2.3 Open-source

One of the key goals of the JASA project is to become a repository for *reference implementations* of various entities that are commonly-used in agent-based computational economics. For example, JASA contains implementations of many common trading strategies that are reported on elsewhere in double-auction simulation experiments. These implementations have undergone testing in the form of replication attempts, and have associated automated regression tests that check whether these components continue to replicate the original work after any changes to the software system are made. By making these implementations publicly and freely available, as early in their development as possible, I have been able to gain invaluable feedback from other researchers as to the correctness of particular re-implementations of other researchers work.

Since many of the components in ACE research are procedural in nature their more concise and accurate description is in some general-purpose programming language. It is my view that pseudo-code is not the best means of expressing these entities; rather, I prefer to use actual runnable code. By using strict coding standards, and object-oriented design techniques, I have been able to develop runnable-code that is just as easy to read as pseudo-code. This has the significant advantage that other researchers can actually execute this code, rather than relying on reverse-engineering of the code in order to study the exact behaviour of the entities I describe.

Visualisation

Providing a visual representation of the state of the simulation can allow for a easier and more institutive analysis, as well as quickly identifying unusual or aberrant behaviour that may be the result of software defects[8]. Figure 5.1 shows a screen-shot of the simulator running with the aide of Repast visualisation features. The simulator allows the auction state tuple $(\hat{MB}, \hat{MB'}, \hat{MS}, \hat{MS'})$ to be plotted graphically in real-time, as well as the true supply and demand schedules (MB, MS, MB', MS'). Visualisation can be turned off when running batch experiments in order to gain maximum time performance.

[8]That is not to say, however, that visualisation is a panacea, or that it is easy to design and implement well in the general case.

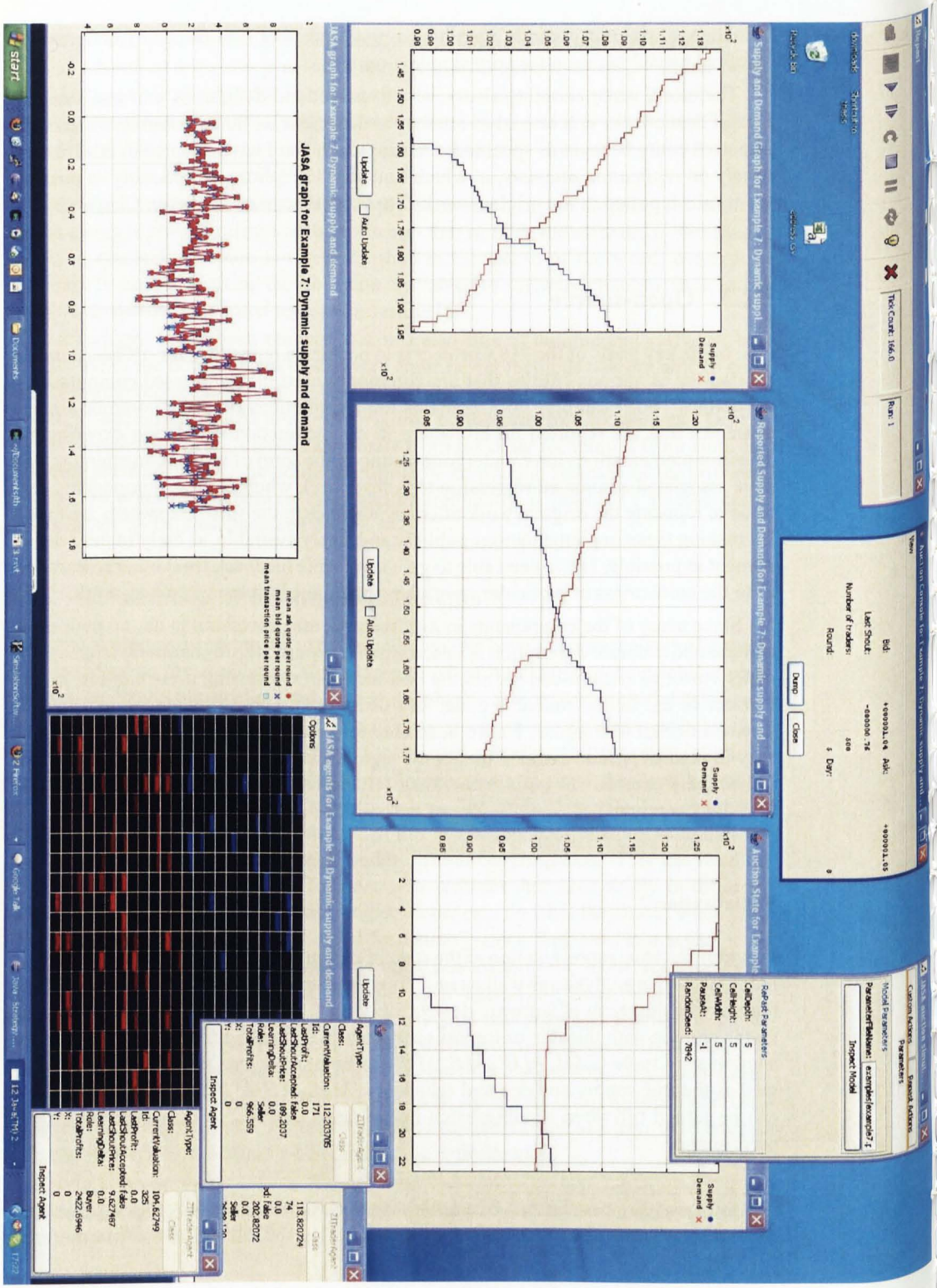

Figure 5.1: Real-time visualisation of auction simulation using RePast.

5.3 Design overview

In this section I give an overview of some of the key components of the auction simulator. The UML diagrams that are referenced from this section can be found in Appendix A.

5.3.1 The shout matching algorithm

As discussed in Section 3.2.6, a core part of simulating *any* auction mechanism is maintaining four sets of shouts: the matched asks ($\hat{MS}$), the matched bids ($\hat{MB}$), the unmatched asks ($\hat{MS}'$), and the unmatched bids ($\hat{MB}'$).

I chose the 4-heap algorithm [159] as the basis of my design, since the 4-heap algorithm specifically addresses both of my design goals. The key innovation of the 4-heap algorithm, as far as performance is concerned, is the use of binary-heaps to maintain the state of the auction; this allows the fundamental operations of an auction: shout insertion and removal, clearing and providing quotes to be carried out very efficiency. Specifically, for a single-unit auction with L active shouts, of which M are asks and N are bids:

- shout insertion/removal can be carried out in $O(ln(L))$ time.
- market quotes[9] can be provided in $O(1)$ time
- clearing[10] can be carried out in $O(min(M, N))$ time.

The 4-heap algorithm is also general; i.e. it is capable of maintaining state for a wide variety of auction mechanisms.

Figure A.1 shows the UML class diagram for the class `FourHeapShoutEngine`. The binary heap attributes `bIn`, `Bout`, `sIn`, `sOut` correspond to $\hat{MB}$, $\hat{MB}'$, $\hat{MS}$, $\hat{MS}'$ respectively, as defined in Section 3.2.6.

The shout-matching service is accessed through the interface `ShoutEngine`, so that alternative matching algorithms can be plugged in if so desired.

5.3.2 Auction mechanisms

The different auction mechanisms are encapsulated through the `Auctioneer` interface, which defines how the clearing operation and quote-generation are scheduled in response to different auction events. See Figure 5.2 for an illustrative example.

Figure A.2 shows a sample of the different double-auction mechanisms that are implemented in `JASA`. Each auctioneer can be configured with a specific class of `PricingPolicy`, the class heirarchy for which is illustrated in Figure A.3. These classes implement the various aspects of our generic double-auction model specified in the Chapter 3. Table 5.3.2 lists the formal specification associated with each class. The experimenter may choose from the existing classes, or they may extend the existing functionality by writing their classes that implement the relevant interface

[9] see Section 3.2.7.
[10] See Section 3.2.9.

Class	*Formal specification*
`ClearingHouseAuctioneer`	Section 3.3
`ContinuousDoubleAuctioneer`	Section 3.4
`DiscriminatoryPricingPolicy`	Section 3.3.2
`InOrderPricingPolicy`	Section 3.3.3
`UniformPricingPolicy`	Section 3.3.1

Table 5.1: Auction rules reference

Class	*Formal specification*
`NPTRothErevLearner`	Section 4.2.4
`RothErevLearner`	Section 4.2.4
`StatelessQLearner`	Section 4.2.4

Table 5.2: Learning algorithm reference

5.3.3 Agents and trading strategies

The strategic behaviour of each agent — the choice of price and quantity at any given time – is decoupled from other aspects of the agent's behaviour such as determining valuation, or replenishing stock levels at the end of a trading day. Thus we have two separate class hierarchies for agents and strategies. Figure A.6 illustrates the `TradingAgent` interface, which is implemented by `AbstractTradingAgent`, each instance of which can be configured with a particular class of trading strategy. The class heirarchy for trading strategies is illustrated in Figure A.5. Each subclass in this heirarchy corresponds to a strategy defined in Chapter 4, and the relevant mappings are shown in Table 5.3.

The decoupling of strategic behaviour from agent house-keeping functionality allows new strategies to be configured via composition. Of particular interest is the `MarkupStrategyDecorator` class, which can be configured to bid a fixed percentage markup on top of another strategy, and the `MixedStrategyClass` class which can be configured to play a number of different "pure" sub-strategies with different probability.

The reinforcement-learning strategies described in section 4.2.4 are implemented by the `StimuliResponseStrategy` class. Strategies of this type can be configured to use different learning algorithms, which are encapsulated in a separate class-heirarchy; Figure A.7 illustrates the relationship between trading strategies and learning algorithms. This design results in minimal dependencies thus allowing the various learning algorithms implemented by JASA to be reused in non-trading contexts.

5.3.4 Events

Figure A.4 illustrates JASA's event architecture. Different types of event are encapsulated in different subclasses of `AuctionEvent` and the various entities in the sim-

```
public class ClearingHouseAuctioneer
        extends TransparentAuctioneer
        implements Serializable {

  protected ZeroFundsAccount account;

  public ClearingHouseAuctioneer() {
    this(null);
  }

  public ClearingHouseAuctioneer( Auction auction ) {
    super(auction);
    setPricingPolicy(new UniformPricingPolicy(0));
    account = new ZeroFundsAccount(this);
  }

  public void generateQuote() {
    currentQuote =
        new MarketQuote(askQuote(), bidQuote());
  }

  public void endOfRoundProcessing() {
    super.endOfRoundProcessing();
    generateQuote();
    clear();
  }

  public void endOfAuctionProcessing() {
    super.endOfAuctionProcessing();
  }

  public Account getAccount() {
    return account;
  }
}
```

Figure 5.2: The source-code for the ClearingHouseAuctioneer class. This code specifies that the quote-generation and clearing operations happen at the end of every round.

Class	*Formal specification*
`EquilibriumPriceStrategy`	Section 4.1.2
`GDStrategy`	Section 4.2.3
`KaplanStrategy`	Section 4.2.2
`RandomConstrainedStrategy`	Section 4.1.4
`StimuliResponseStrategy`	Section 4.2.4
`TruthTellingStrategy`	Section 4.1.1
`ZIPStrategy`	Section 4.2.1

Table 5.3: Trading strategy reference

ulator that respond to auction events do so through the `AuctionEventListener` interface.

5.4 Summary and contribution

As we saw in Chapter 3 many of the variants of the double-auction mechanism are difficult to analyse using conventional analytical tools. Therefore, the approach to mechanism design I take in this thesis is an empirical one, in which real-life observations and simulation play a key role. In this chapter I have given an overview of the simulation software — JASA — that I developed in order to conduct the experiments in this thesis. Since these experiments are a key part of my research, it is important that their implementation in sofware is as readable and transparent as possible, and is designed according to a sound methodology in an attempt to ensure its correctness.

In this chapter I have described how JASA was developed using best-practice engineering methodology for agent-based simulation: it is object-oriented, extensible, configurable, high-performance and open-source. I have used principles of *agile software engineering* in keeping with the dynamic nature of software designed for research purposes, specifically: automated unit-testing, early and frequent releases to other users, community bug-tracking and an emphasis on collaborative software development. JASA has undergone many refinements and bug-fixes throughout its development and is now of sufficient maturity that it is used by several different research teams around the world for research into agent-based computational economics [140, 60, 99, 16, 96].

Chapter 6

Replication Experiments

6.1 Introduction

In the previous chapter I discussed the fact that replication experiments are a key technique in validating software used for agent-based computational economics. In this chapter I report on the most important replication experiments that were used to validate the software used throughout this thesis.

6.2 Control Experiments

In many of the experiments discussed in this thesis the main variable that is measured is the efficiency of the market EA (defined by equation 3.6). As discussed in Chapter 4 there are two *control* strategies that should yield $EA = 1$ in a wide variety of circumstances: the TT strategy (Section 4.1.1) and the EPS strategy (Section 4.1.2). The former should robustly yield $EA = 1$ in a CH mechanism provided that every agent uses the TT strategy, the latter similarly in a CDA mechanism. This suggests that experiments with agents using these strategies can serve as important controls.

The TT strategy is tested in a CH with a $k = 0.5$ discriminatory-pricing policy (Section 3.3.2). Agents' valuations are drawn from a uniform distribution on the interval $[50, 100]$ and the market is run for 10^2 rounds. This experiment is run 10^2 times, and if $EA \neq 1$ for any iteration then an error is reported.

The EPS strategy is tested in a CDA with a $k = 0.5$ discriminary-pricing policy. Agents' valuations are drawn from a uniform distribution on the interval $[50, 300]$ and the market is run for 10^2 trading days, each of which lasts 20 rounds:

$$eor_{t-1} \in E_{t-1} \implies NR_t = NR_{t-1} + 1 \tag{6.1}$$

$$NR_{t-1} \mod 20 = 0 \implies eod_t \in E_t \tag{6.2}$$

This experiment is repeated 2×10^2 times, and if $EA \neq 1$ for any iteration then an error is reported.

Both of these experiments are part of the automated regression testing suite which is run whenever changes to the simulation code are made. Thus the results of these control experiments are implicit in any experimental results reported in this thesis.

6.3 Nicolaisen's Electricity Market

Nicolaisen, Petrov and Tesfatsion [98] (abbreviation NPT) describe several experiments using a multi-unit clearing-house auction[1] with discriminatory $k = 0.5$ pricing[2]. The domain they study is that of market design for deregulated electricity markets, in which small numbers of traders with relatively static valuations repeatedly interact with each other over a long time period.

Nicolaisen *et al.* were concerned with market-power effects as the number of traders on the supply side or the demand side varied; that is, to what extent does the market favour buyers or sellers as each group becomes smaller, and thus in more of a monopoly-like situation. N_S and N_B denote the number of sellers and the number of buyers respectively. As in the original NPT paper, in the scenarios we consider we examine cases where $N_S = 3$, $N_S = 6$, $N_B = 3$, $N_B = 6$ and all corresponding combinations. Buyer valuations are taken from the multisets:

$$VB = \{37, 37, 17, 17, 12, 12\} \iff N_B = 6 \tag{6.3}$$

$$VB = \{37, 17, 12\} \iff N_B = 3 \tag{6.4}$$

Correspondingly for seller valuations:

$$VS = \{11, 11, 16, 16, 35, 35\} \iff N_S = 6 \tag{6.5}$$

$$VS = \{11, 16, 35\} \iff N_S = 3 \tag{6.6}$$

Each group of agents has a fixed, finite generating capacity that determines the maximum amount of electricity resource that they are capable of trading at any given time. The variable CS denotes the generating capacity of sellers and the variable CB denotes the generating capacity of buyers. Agents place shouts at quantity equal to their generating capacity, thus we modify equations 4.1 and 4.2 to incorporate multi-unit bidding according to generating capacity:

$$Z(i,t) = (bid, a_i, \zeta(i,t), CB, t) \iff a_i \in B \wedge a_i \in K_t \tag{6.7}$$

$$Z(i,t) = (ask, a_i, \zeta(i,t), CS, t) \iff a_i \in S \wedge a_i \in K_t \tag{6.8}$$

Agents use the modified version of the Roth-Erev trading strategy specified in Section 4.2.4; thus $\zeta(i,t)$ is given by equations 4.51 and 4.52, and:

$$RL_{\lambda_i}(t) = RE'(i,t)\ \forall i \tag{6.9}$$

[1] See section 3.3

[2] See section 3.3.2

where $RE'(i,t)$ is defined as in Section 4.2.4. Each agent's strategy is configured with the following parameters which are taken from the "best-fit" parameter set used by NPT [98, p. 10].

$$\begin{aligned} RE_{\epsilon_i} &= 0.2 \quad \forall_i \\ RE_{\rho_i} &= 0.1 \quad \forall_i \\ RE_{s_i} &= 9 \quad \forall_i \\ RE_{k_i} &= 100 \quad \forall_i \end{aligned}$$

Nicolaisen *et al.* tested a number of different scenarios by systematically varying the relative concentration of sellers to buyers, $RCON$, and the relative generating capacity of buyers to sellers, $RCAP$, where these are defined:

$$RCON = \frac{N_S}{N_B} \tag{6.10}$$

$$RCAP = \frac{N_B \times CB}{N_S \times CS} \tag{6.11}$$

The outcomes of interest are the market-power available to the buyers and sellers respectively, denoted by the variables MPB and MPS respectively, and defined by the equations:

$$MPB = \frac{PBA - PBCE}{PBCE} \tag{6.12}$$

$$MPS = \frac{PBS - PSCE}{PSCE} \tag{6.13}$$

where PBA and PSA denote the profits of the buyers verses sellers once the auction has finished ($t = t'$):

$$PBA = \sum_{\forall a_i \in B} \Gamma_{t'}(a_i) \tag{6.14}$$

$$PSA = \sum_{\forall a_i \in S} \Gamma_{t'}(a_i) \tag{6.15}$$

and the variables $PBCE$ and $PSCE$ denote the profits of buyers and sellers respectively in *competitive equilibrium*. This is calculated by running a control experiment in which all agents use the EPS strategy as defined in Section 4.1.2, and then calculating $PBCE$ as per PBA (similarly $PSCE$ is calculated as per PSA).

The efficiency of the outcome is denoted EA' and defined:

$$EA' = \frac{PBA + PSA}{PBCE + PSCE} \times 100 \tag{6.16}$$

This is simply the efficiency EA defined by equation 3.6 expressed as a percentage; that is:

$$EA' = EA \times 100 \tag{6.17}$$

Results

RCON		RCAP $\frac{1}{2}$		RCAP 1		RCAP 2	
2	EA'	99.76	(0.84)	99.95	(0.20)	99.60	(3.4)
	MPB	-0.36	(0.17)	0.06	(0.30)	0.01	(0.45)
	MPS	1.53	(0.73)	-0.05	(0.27)	-0.02	(0.40)
1	EA'	99.07	(2.87)	99.61	(0.40)	98.27	(6.16)
	MPB	-0.27	(0.14)	-0.37	(0.19)	0.12	(0.38)
	MPS	1.13	(0.58)	1.26	(0.66)	-0.14	(0.36)
$\frac{1}{2}$	EA'	96.66	(6.18)	99.64	(0.35)	99.98	(0.01)
	MPB	-0.41	(0.15)	-0.37	(0.20)	-0.13	(0.28)
	MPS	1.58	(0.58)	1.28	(0.68)	0.11	(0.25)

Table 6.1: Replicated results for 10^4 auction rounds

RCON		RCAP $\frac{1}{2}$		RCAP 1		RCAP 2	
2	EA'	100.00	(0.00)	99.49	(0.01)	100.00	(0.00)
	MPB	-0.04	(0.07)	-0.07	(0.26)	-0.07	(0.24)
	MPS	0.19	(0.32)	0.21	(0.19)	-0.06	(0.19)
1	EA'	94.13	(0.09)	99.66	(0.01)	100.00	(0.00)
	MPB	-0.16	(0.09)	-0.08	(0.07)	0.06	(0.24)
	MPS	0.60	(0.38)	0.22	(0.28)	-0.05	(0.19)
$\frac{1}{2}$	EA'	95.22	(0.09)	99.56	(0.01)	100.00	(0.00)
	MPB	-0.14	(0.07)	-0.06	(0.05)	0.10	(0.20)
	MPS	0.59	(0.36)	0.20	(0.19)	-0.08	(0.16)

Table 6.2: Nicolaisen et al.s' results for 10^4 auction rounds

Nicolaisen *et al.*'s original results are re-presented in Table 6.2. The results that I obtain using the current version of my simulation framework are shown in Table 6.1. Each value reported is the mean from $n = 100$ samples of the experiment with the standard deviation presented in brackets[3]. These were obtained using the 64-bit version of the Mersenne Twister PRNG [85] using double-precision IEEE 754 floating point arithmetic [135].

Although the results are not numerically identical to those in the original experiment, the *qualitative* outcomes of both the original and the replicated experiments are very similar. In particular, there are relatively few sign discrepancies between the market-power outcomes when comparing the replicated results with the original results, and where there is a sign discrepancy the absolute value of the market-power variable is closer to zero than when there is not. This is the same qualitative criteria

[3]Note that the standard *error* of the mean is given by $\sigma_M = \frac{\sigma}{\sqrt{n}}$ which is not what is directly reported in the results table

that Nicolaisen et al. use to compare their experimental outcomes with the analytically predicted outcomes for market power.

6.4 Cliff's Zero-Intelligence Plus Strategy

As discussed in Section 4.2.1, Cliff [28] set out to explore an apparent anomaly in the work of Gode and Sunder [58]. Gode and Sunder demonstrated that their zero-intelligence constrained strategy (Section 4.1.4) was able to yield high efficiency outcomes in a CDA mechanism that were comparable to those of human agents. However, in analysing the micro-behaviour of the zero-intelligence agents compared with the human trading behaviour, Cliff observed that the statistical distribution of transaction prices around the equilibrium price would become significantly greater if agents' valuations were randomly assigned from different distributions to those of Gode and Sunder, and thus he argued that the ZIC strategy was not an adequate model of human trading behaviour. This is because in these scenarios, although

> *"As with the ZI-C traders, measures of allocative efficiency for ZIP traders are typically very high ..."* [28, p. 47]

when analysing the statistical deviation of transaction prices from the equilibrium price $p*$ an alternative model, zero-intelligence *plus* is required in order to replicate this micro-behaviour in a wider range of circumstances:

> *"...the data in these graphs serves to demonstrate that simple ZIP trading strategies can readily achieve results that are impossible when using ZI-C traders, and are closer to those expected from human subjects... on these ground at least, the minimally adaptive ZIP traders represent a significant advance on the work of Gode and Sunder."* [28, p. 46]

The metric of interest here is the root mean square (RMS) of the difference between observed transaction prices and the equilibrium price, denoted α, and defined formally as:

$$\alpha_t = \frac{100\sigma_t}{p*} \tag{6.18}$$

where σ_t is defined:

$$\sigma_t = \frac{\sqrt{\sum_{c \in C_t}(c_p - p*)^2}}{|C_t|} \tag{6.19}$$

The final outcome is averaged over the duration of the entire experiment:

$$\alpha = \frac{\sum_{t=0}^{t=t'} \alpha_t}{t'} \tag{6.20}$$

In these experiments agents are endowed with a trade entitlement for a single unit of commodity which is replenished at the start of each trading day, and decremented each time they enter into a transaction. Let $T_{(i,t)}$ denote the trade entitlement for agent a_i at time t. Agents place bids only if they are entitled to trade:

$$Z(i,t) = (bid, a_i, \zeta(i,t), T_{(i,t)}, t) \iff a_i \in B \wedge a_i \in K_t \wedge T_{(i,t)} > 0$$
$$Z(i,t) = (ask, a_i, \zeta(i,t), T_{(i,t)}, t) \iff a_i \in S \wedge a_i \in K_t \wedge T_{(i,t)} > 0$$

The trade entitlement is reset at the start of each day:

$$eod_{t-1} \in E_{t-1} \implies T_{(i,t)} = 1 \; \forall i \tag{6.21}$$

and decremented for each transaction:

$$\exists c \in C_{t-1} \implies T_{(agent(c),t)} = 0 \tag{6.22}$$

Parameters

The auction was run for 20 days, each consisting of 50 rounds.

$$eor_{t-1} \in E_{t-1} \implies NR_t = NR_{t-1} + 1 \tag{6.23}$$

$$NR_{t-1} \mod 50 = 0 \implies eod_t \in E_t \tag{6.24}$$

All experiments were run with a CDA clearing-rule with in-order discriminatory pricing (Section 3.3.3).

The following parameters were used for the ZIP strategy:

$$\begin{aligned} \forall_i \; ZIP_{\sigma_i} &= 0.05 \\ \forall_i \; ZIP_{\alpha_i} &= 0.05 \\ \forall_i \; ZIP_{\lambda_i} &= 0.1 \\ \forall_i \; ZIP_{\mu_i} &= 0.05 \end{aligned}$$

This was compared with a population of agents equipped with the ZIC strategy (Section 4.1.4), configured:

$$\forall_i \; ZIC_{E_i} = 50 \tag{6.25}$$

For each repetition of the experiment agents' valuations were drawn:

$$v_i \sim U(50, 200) \tag{6.26}$$

All floating point computations were performed using IEEE 754 double-precision arithmetic [135] and the 64-bit Mersenne Twister PRNG [85] was used for all random variables.

Results

Table 6.3 shows the mean and standard deviation for α and EA for 50 repetitions of the experiment with different valuations. As is clear, these results are statistically significant since:

$\sigma_M \approx \frac{5}{\sqrt{50}} \approx 0.7$

Thus we see that once we compare ZIP and ZIC under a wide variety of different supply and demand schedules, ZIP achieves higher EA and lower α as reported by Cliff and Bruten.

Strategy	α		EA	
ZIC	17.68	(5.58)	96.40	(2.00)
ZIP	7.18	(2.81)	98.58	(1.40)

Table 6.3: Replication results comparing mean outcomes for ZIC verses ZIP over 50 samples. The standard deviation is shown in brackets.

Acknowledgments

The contribution of others has been particularly important in establishing these results, and so I shall repeat acknowledgment of their contribution in this section. I am particularly indebted to Jinzhong Nui and Kai Cai of the City University of New York, whose collaboration ultimately led to the successful replication of ZIP outcomes.

The NPT results were obtained after much discussion via email with one of the original authors — Leigh Tesfatsion — and I was able to find and fix many bugs in my simulation software as a result of her insights. Additionally Igor A. Walter pointed out a significant bug in the random permutation algorithm used to randomly re-order the order in which agents are polled, which was subsequently fixed.

6.5 Summary and Contribution

Replication attempts are a key part of validating the software used in agent-based computational economics. In this chapter I have reported on control and replication experiments involving key combinations of certain strategies and auction mechanisms. This will provide more confidence in the results reported in later chapters.

Chapter 7

Empirical Game Theory

The automatic discovery of game-playing strategies has long been considered a central problem in Artificial Intelligence. 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. 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 chapter I explore some of the limitations of current co-evolutionary algorithms, and introduce a field known as *empirical game theory* which combines game-theoretic analysis together with simulation methods.

7.1 Nash Equilibrium

The failure of certain types of co-evolutionary algorithms to converge on robust strategies in certain scenarios is well known [148, 45, 20], 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* [132] 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

[1] Or sometimes several populations.

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 [95]. Additionally, if we know the entire set of strategies and payoffs, we can deterministically compute the Nash strategies. If only a single equilibrium exists for a given game, this means that, in theory at least, we can always compute the "appropriate" strategy for a given game.

Note, however, that the Nash strategy is not always the *best* strategy to play in all circumstances. For 2-player zero-sum games, one can show that the Nash strategy is not exploitable. However, if our opponents do not play their Nash strategy, then there may be other non-Nash strategies that are better at exploiting off-equilibrium players. Additionally, many equilibria may exist and in n-player non-constant-sum games it may be necessary for agents to *coordinate* on the same equilibrium if their strategy is to remain secure against exploitation; if we were to play a Nash strategy from one equilibrium whilst our opponents play a strategy from an alternative equilibrium we may well find that our payoff is significantly lower than if we had coordinated on the same equilibrium as our opponents.

7.2 Beyond Nash equilibrium

Standard game theory does not tell us which of the many possible Nash strategies our opponents are likely to play. *Evolutionary* game theory [86] 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 to repeated observations of their own and others' payoffs. One approach to evolutionary game-theory uses the *replicator dynamics* equation to specify the frequency with which different pure strategies should be played depending on our opponent's strategy:

$$\dot{m}_j = [u(e_j, \vec{m}) - u(\vec{m}, \vec{m})]\, m_j \tag{7.1}$$

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.

For any initial mixed-strategy we can find the eventual outcome from this co-evolutionary process by solving $\dot{m}_j = 0$ for all j to find the final mixed-strategy of the converged population. 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 [83] and interior limit states are also Nash equilibria [149, 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 7.1. 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 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.

7.3 Co-evolution

But what if we were to approximate the replicator dynamics by using an evolutionary search over the strategy space? Rather than considering an infinite population consisting of a mixture of all possible pure strategies, we use 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 proportion of the population playing different strategies serves a dual role in a co-evolutionary algorithm [47]. 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

[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.

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 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 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 [46].

These problems have led researchers in co-evolutionary computing to design new algorithms employing game-theoretic solution concepts [44]. In particular, Ficici and Pollack [47] 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 thesis, I address n-player non-constant-sum multi-state games with type-dependent payoffs. In such games, playing the Nash strategy (or an approximation thereof) does not guarantee a participant security against exploitation, thus if there are multiple equilibria, it may be more appropriate to play a *best-response* to the strategies that we infer are in play.

7.4 Empirical Game-Theory

Reeves *et al.* [30] and Walsh *et al.* [145] 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 games, unsurprisingly none of the strategies commonly in use is dominant over the others. Given the lack 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 (eg 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 entry 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

$$s = \frac{(n+j-1)!}{n!(j-1)!} \tag{7.2}$$

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.

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.

The equilibria solutions that are thus obtained are not rigorous Nash equilibria for the full multi-state game; there is always the possibility that an unconsidered strategy could invade the equilibrium. Nevertheless, heuristic-strategy equilibria are more plausible as models of real-world game playing than those obtained using a standard co-evolutionary search precisely because they *restrict* attention to strategies that are commonly known and are in common use. We can therefore be confident that no commonly known strategy for the game at hand will break our equilibrium, and thus the equilibrium stands at least some chance of persisting in the short term future. I will return to this issue in chapter 9. Meanwhile, in the next chapter, we will use heuristic-strategy approximation to analyze two different variants of the double-auction from a design perspective.

Chapter 8

Analysing Auction Mechanisms

In this chapter I will analyze two variants of the double auction market—the clearing house auction and the continuous double auction. The complexity of these institutions is such that they are extremely hard to analyse using traditional game-theoretic techniques, and so I shall use the heuristic-strategy approximation technique described in the previous chapter in order to provide an approximated game-theoretic analysis. As well as finding heuristic-strategy equilibria for these mechanisms, I shall subject them to an evolutionary game-theoretic analysis which will quantify which equilibria are more likely to occur. We can then weight the design objectives for each mechanism according to the probability distribution over equilibria, which will allow us to provide more realistic estimates for the efficiency of each mechanism.

8.1 The CH versus the CDA

In a typical exchange, the market institution attempts to match offers to buy with offers to sell in such a way that the overall surplus extracted from the market is maximized. If offers are considered as signals of agents' valuations for a resource, and assuming agents signal truthfully, then an auctioneer can maximize allocative efficiency by matching the highest buy offers with the lowest sell offers. In this chapter I compare two types of exchange:

- A $k = 0.5$ continuous double-auction (CDA) market in which trades are executed as new offers arrive and prices are set half-way between the bid and ask price, as described in Section 3.4; and
- A discriminatory price $k = 0.5$ clearing-house (CH), as described in Sections 3.3 and 3.3.2, in which the auctioneer waits for all traders to place offers before clearing the market.

On casual inspection of the CDA, we might expect that it is designed according to auction-theory principles, and so should maximize allocative efficiency when agents signal truthfully (see 4.1.1). Surprisingly, however, it turns out that surplus extraction

in a CDA is extremely *poor* under truth-telling —typical values are $EA \approx 0.80$, which is extremely low compared with outcomes of almost $EA \approx 0.98$ which are observed with the non-truthful strategies that are actually adopted by human traders [58].

As we saw in Section 3.4, the reason for this poor efficiency is easy to spot; the continuous clearing rule results in myopic matching. When the clearing operation is performed the auctioneer has only a partial view of the aggregate supply and demand in the market place. In order to maintain a high throughput of actual transactions, the auctioneer impatiently clears the market before every trader has the opportunity to place their bid. However, as we saw in Section 6.4, the extremely surprising thing about this institution is that rational agents acting locally to maximize their own profit are able to compensate for this efficiency loss by placing extra-marginal, non-truthful bids, which collectively result in high-efficiency outcomes.

Much analysis of the CDA has focused on showing that although the CDA is not an incentive-compatible mechanism, it can be considered "almost incentive-compatible" by virtue of the fact that trading strategies with only a minimal amount of intelligence are able to extract high surpluses from the market [58, 28]. However, such approaches are insufficient for market-design purposes, because they fail to demonstrate that such minimalist strategies are *dominant* against more sophisticated strategies. For example, if we decide to use a population of homogeneous ZIP traders to ascertain how the CDA and the CH markets compare with each other, we are making an implicit assumption that the state of affairs whereby all agents adopt the ZIP strategy is an equilibrium state. However, in order to justify this assumption we should ensure that any hypothetical equilibrium of ZIC or ZIP traders is not susceptible to invasion by an alternative strategy

Ideally, we would like to find the game-theoretic solution for the CDA, and show that although truth-telling or other minimalist strategies are not dominant, we can still find the theoretical mix of strategies that are best-responses to each other, and demonstrate that the institution performs well in game-theoretic equilibria. However, even at this point, the CDA along with other variants of the double-auction market, confounds auction theorists by admitting of no unequivocal equilibrium solution[1].

Hence in the absence of robust analytical tools, much analysis of this institution has used an ad-hoc mixture of computer simulation and laboratory experiments [51]. These techniques are invaluable, since they are able to faithfully incorporate many of the complex details of the market institution which lead to intractability under conventional analysis. However, the results thus obtained are often critised for being difficult to generalize in the absence of compelling models that explain the observed outcomes.

However, as discussed in chapter 7, techniques have been developed recently that combine simulation-based approaches with an approximated game-theoretic analysis. In the following sections, I describe in detail an empirical game-theoretic analysis of the CDA and the CH mechanisms.

[1]That is, in which *all* equilibrium strategy profiles are clearly identified. The relavant literature is reviewed in Chapter 2.

8.2 Experimental setup

In order to compare the CDA and CH, we must first generate a heuristic payoff matrix for each institution by sampling many simulations of the market game. As in [145], at the start of each game half the agents are randomly assigned to be buyers and the remainder are chosen as sellers. For each run of the game, valuations are drawn as in [145]:

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

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. 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 [85] was used to draw all random values used in the simulation and all floating point calculations were performed using IEEE 754 double-precision arithmetic [135]. Each entry in the heuristic payoff matrix was computed by averaging the payoff to each strategy across 10^4 simulations.

8.2.1 Choice of heuristic strategies

In choosing candidate heuristic-strategies for our analysis, we need to consider the following constraints:

1. The strategies chosen should be able to trade in both types of mechanism.
2. They should be representative of strategies that are commonly known for these types of mechanism.
3. We should include the truth-telling strategy (TT), since we are interested in the incentive-compatibility properties of each mechanism.

Accordingly, I chose the strategies TT, RE, TK and GD as described in table 8.1: the TT strategy was chosen in accordance with constraint 3 above; the TK strategy was chosen since it is a very simple strategy that was also the winner of the original Santa-Fe trading strategy competition [51] and is prevalent in on-line single-sided auctions [120]; 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 [119]:

Abbreviation	*Description*
TT	The truth-telling strategy, (section 4.1.1).
RE	The reinforcement-learning strategy (section 4.2.4), configured with the Roth-Erev learning algorithm (section 4.2.4).
TK	Todd Kaplan's sniping strategy (section 4.2.2) which waits until the last minute before placing a shout.
GD	The Gerstad-Dickhaut strategy (section 4.2.3) which estimates the probability of shouts being accepted as a function of price and bids to maximise expected payoff.

Table 8.1: The heuristic strategies chosen for the analysis

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

8.2.2 Choice of market size

Auction marketplaces with a small, fixed, number of traders who repeatedly interact are becoming more common place with the advent of business-to-business electronic commerce and the deregulation of wholesale markets such as electricity [98]. As discussed in sections 1.1 and 5.1.3, these are the most difficult scenarios to evaluate analytically using conventional techniques. With large numbers of agents the market starts to approximate the continuous case; as $|A| \rightarrow \infty$ the supply and demand schedules start to approximate smooth curves, as will the reported supply and demand $\hat{MB}$ and $\hat{MS}$, and it is very likely that $\max(MS) \approx \max(\hat{MS})$ and $\min(MB) \approx \min(\hat{MB})$ regardless of which agent plays which strategy. Hence, in a CH for example:

$$\frac{e\hat{q}_a - e\hat{q}_b}{2} \approx p*$$

and so we would clear at close to an equilibrium price regardless of strategy choice[2]. In other words, we can use general equilibrium theory to predict the likely outcome. However, when we have small numbers of agents, the system becomes more discrete and unpredictable, and we have to pay much more attention to the behaviour of the

[2]This argument is merely a sketch, however, see [53] for a more rigorous example of how tractable solutions emerge when the number of agents is very large

individual components (agents) in order to predict outcomes[3]. Given that analytic approaches are generally intractable for scenarios with small $|A|$, this is my justification for using an *empirical* game-theoretic analysis, and I will analyze mechanisms with $|A| = 4$, $|A| = 6$ and $|A| = 12$ traders.

8.3 Dynamic Analysis

Once the heuristic payoff matrix has been computed, we can subject it to a game-theoretic analysis. In conventional mechanism design, 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 real-world design problems. Since our design objectives depend on outcomes, we should give more consideration to outcomes that are more likely than low probability outcomes. For example, 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 [145], we will use *evolutionary* game-theory [86] 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 7.1), 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 triangle embedded in the three dimensional plane 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

[3] As Gintis points out [56], this is analogous to the modelling of physical systems at different scales. In large-scale systems we can model bodies as homogeneous collections of simple particles whose macro-behaviour is the statistical ensemble of many simple micro-interactions yielding Newtonian mechanics. However, when we analyse behaviour at the molecular and subatomic scales, the characteristics of individual particles play a more prominent role and we get correspondingly more complicated and discrete models (chemistry and quantum mechanics). In this analogy, ACE is to general equilibrium theory as Newtonian mechanics is to quantum mechanics.

[4] See [149, pp. 3–7] for a more detailed exposition of the geometry of mixed-strategy spaces.

converged population. As discussed in Section 7.2, this has a significant advantage over non-game-theoretic co-evolutionary search, such as [65], in that we can *guarantee* [149, 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 [83] are Nash equilibria.

Thus the Nash equilibrium solutions are embedded in the stationary points of the direction field of the dynamics specified by equation 7.1. 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 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 [18]. This intuitive definition of basin size is formalized as follows. Let the function

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

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

$$T(\vec{x}, M \subset \triangle^n) = |\{\vec{y} : \vec{y} \in M \wedge \vec{m}(0) = \vec{y} \wedge \exists t\ \vec{m}(t) = \vec{x} \wedge \dot{m}(t) = 0\}| \quad (8.1)$$

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 $\vec{x}$:

$$\beta(\vec{x}, M) = \frac{T(\vec{x}, M)}{|M|} \quad (8.2)$$

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:

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

[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.

8.4 Results

Since the vector $\vec{m}$ in equation 7.1 represents a mixed-strategy, that is, a discrete probability distribution, we have $\sum m_i = 1$; thus each $\vec{m}$ lies in the unit-simplex (p. 89). For $n = 3$ strategies we can project the unit-simplex $\triangle^3$ onto a two dimensional triangle whose vertices correspond to the pure strategies $(1, 0, 0)$, $(0, 1, 0)$, and $(0, 0, 1)$. By plotting the time-evolution of equation 7.1 we can then identify the switching between strategies. Figure 8.1 shows the direction-field when we consider evolutionary switching between the three strategies TT, RE and GD, in a CH market populated by $|A| = 12$ agents which are selected at random from a larger population of traders on each play of the game.

The direction field gives us a map which shows the trajectories of strategies of learning agents engaged in repeated interactions, from a random starting position. Thus, for Figure 8.1, each agent participant has a starting choice of 3 pure strategies (TT, RE and GD) and any mixed (probabilistic) combination of these three. The pure strategies are indicated by the 3 vertices of the simplex (triangle), while mixed strategies are indicated by points on the boundaries or in the middle of the simplex.

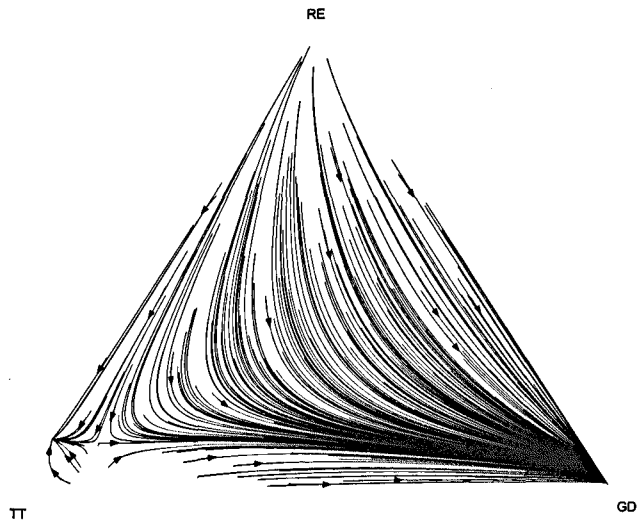

Figure 8.1: 3-dimensional replicator dynamics direction field for a 12-agent clearing-house auction with the three strategies RE, TT and GD.

An agent in the population at large is assigned a pure strategy randomly chosen from the set $\{\mathrm{TT}, \mathrm{RE}, \mathrm{GD}\}$ to start, but switches to an alternative strategy with probability proportional to the relative payoffs observed from agents playing alternative strategies. Thus in a large population of agents, the proportion of agents playing each pure strategy will vary according to the learning process described by Equation 7.1. The paths shown in Figure 8.1 trace this sequence of adjustments. Since at the beginning of each market game, each agent in the smaller population of $|A| = 12$ agents is chosen at random from the larger population of agents playing pure strategies, we

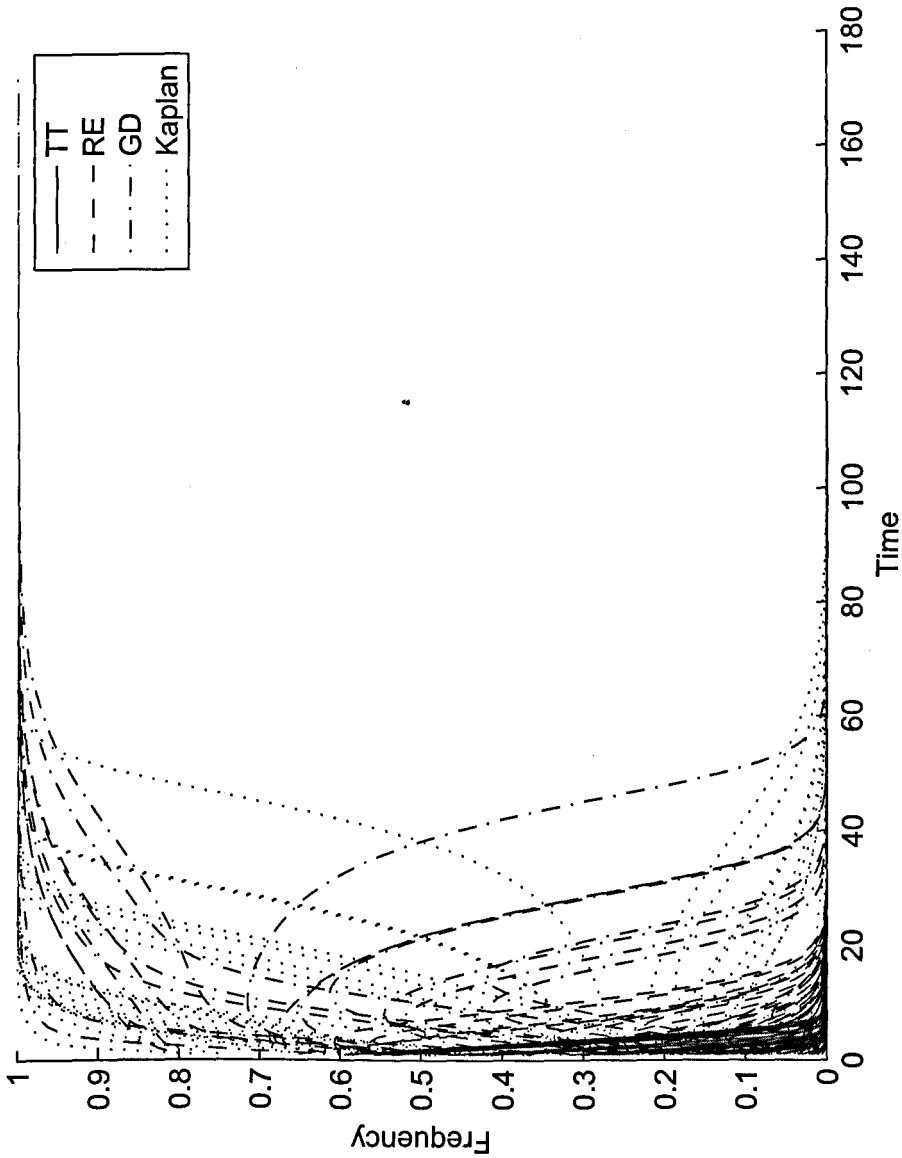

Figure 8.2: Replicator dynamics as time-series for CH with 4 agents

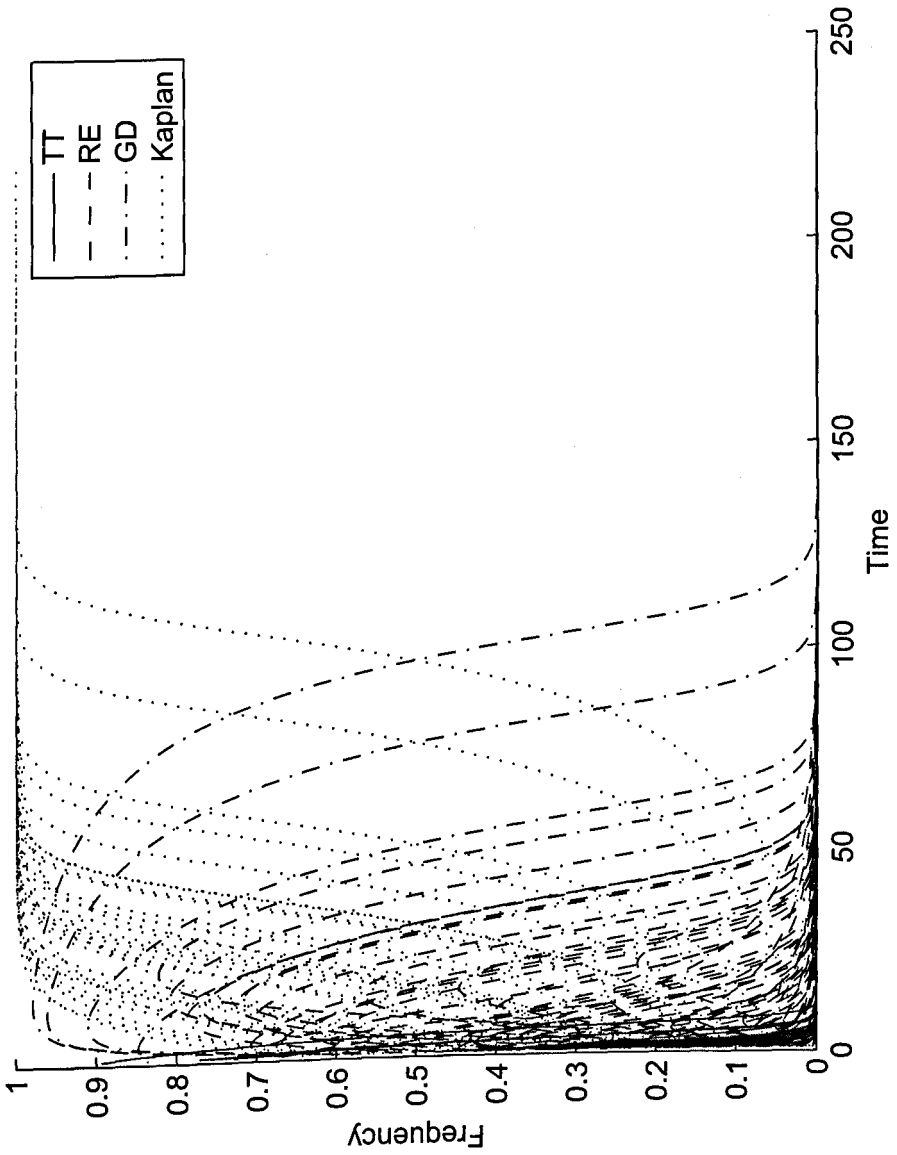

Figure 8.3: Replicator dynamics as time-series for CDA with 4 agents

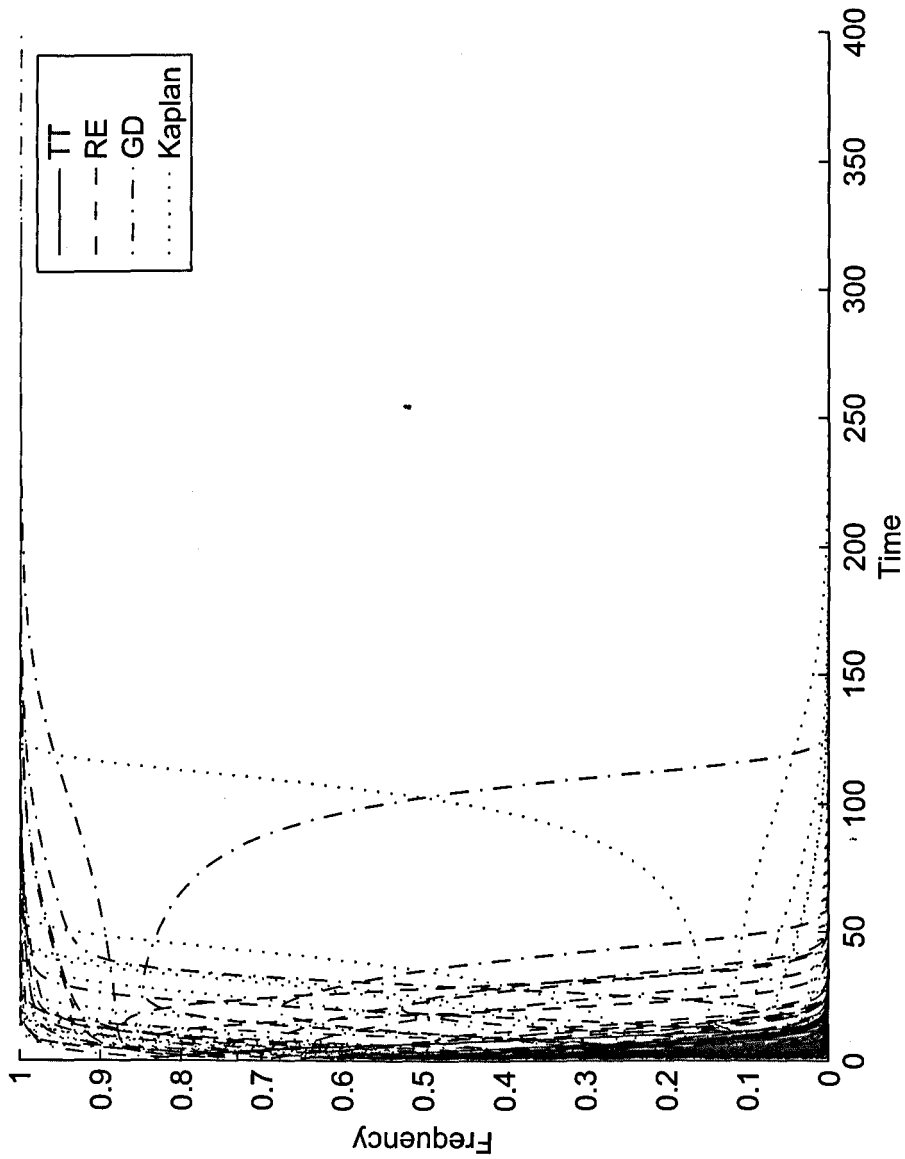

Figure 8.4: Replicator dynamics as time-series for CH with 6 agents

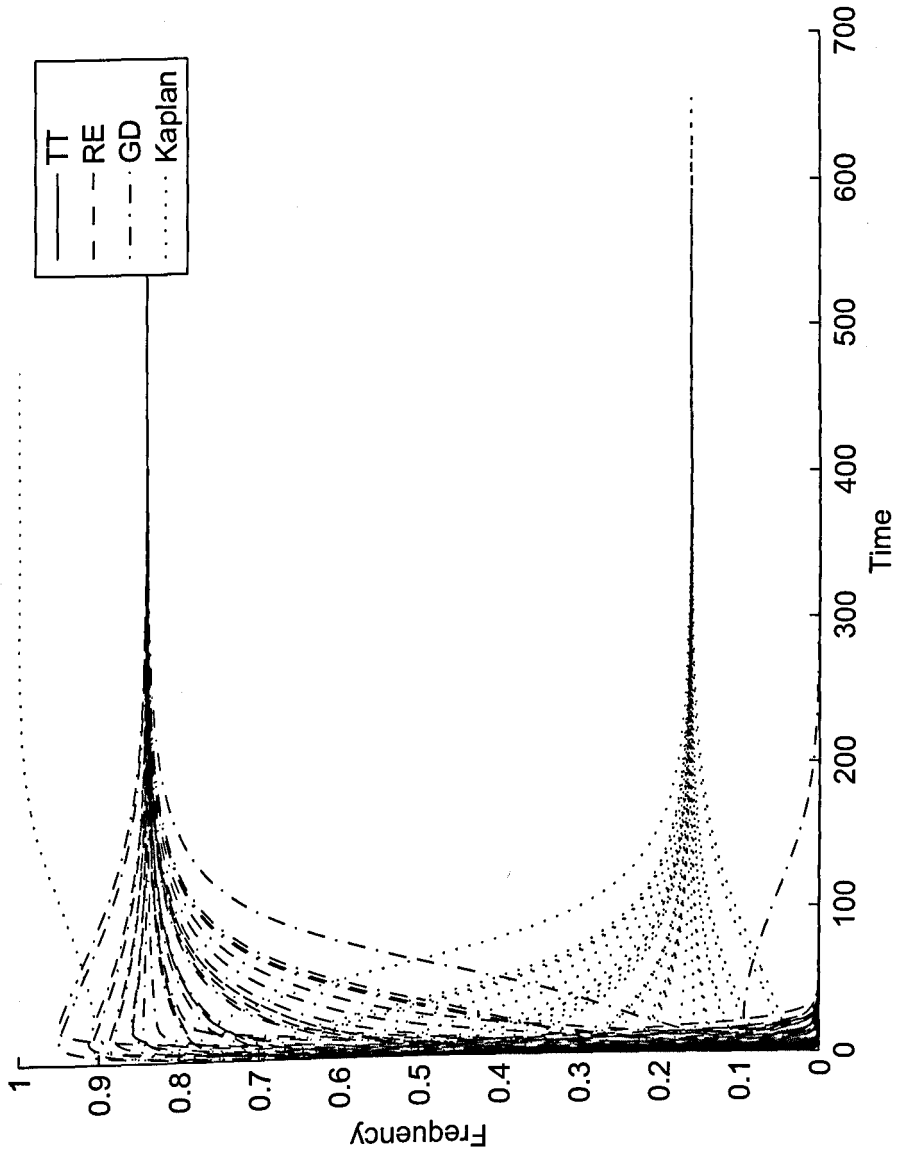

Figure 8.5: Replicator dynamics as time-series for CDA with 6 agents

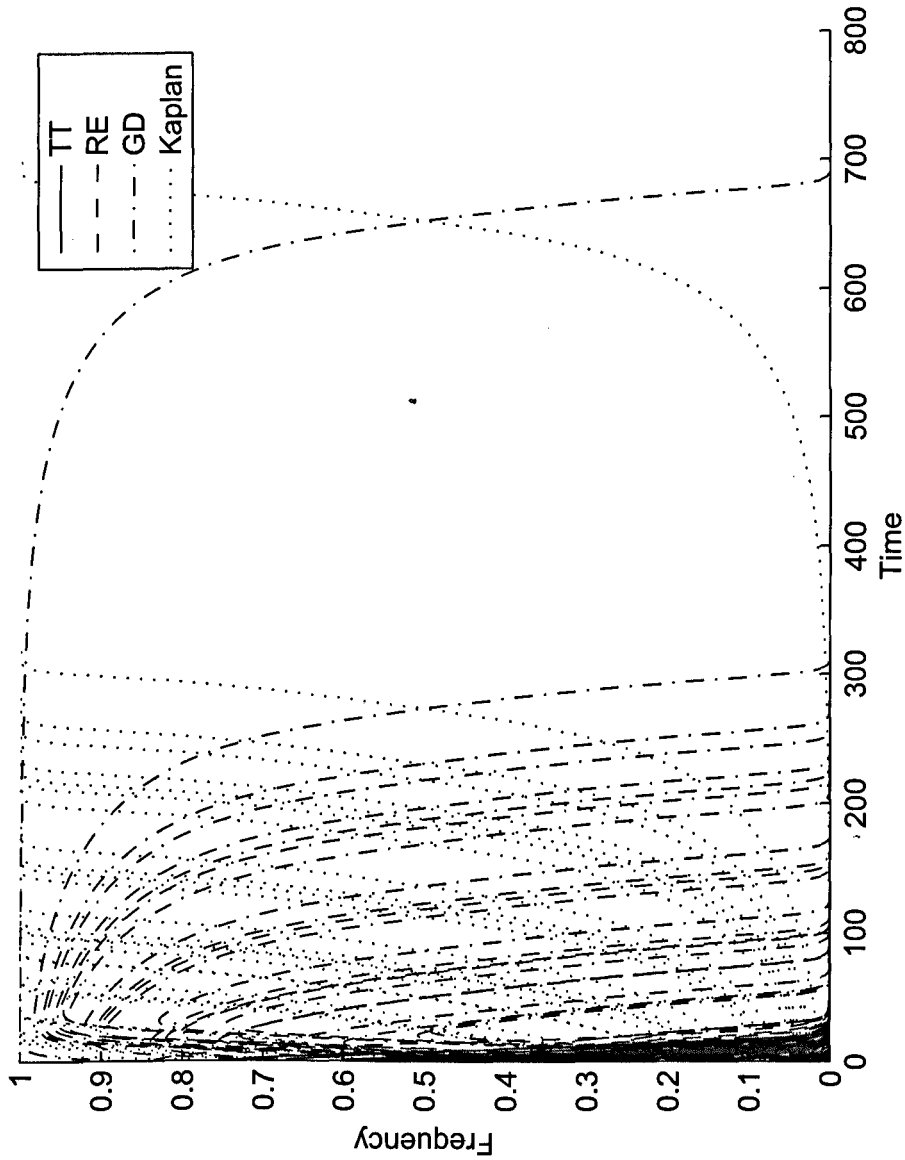

Figure 8.6: Replicator dynamics as time-series for CH with 12 agents

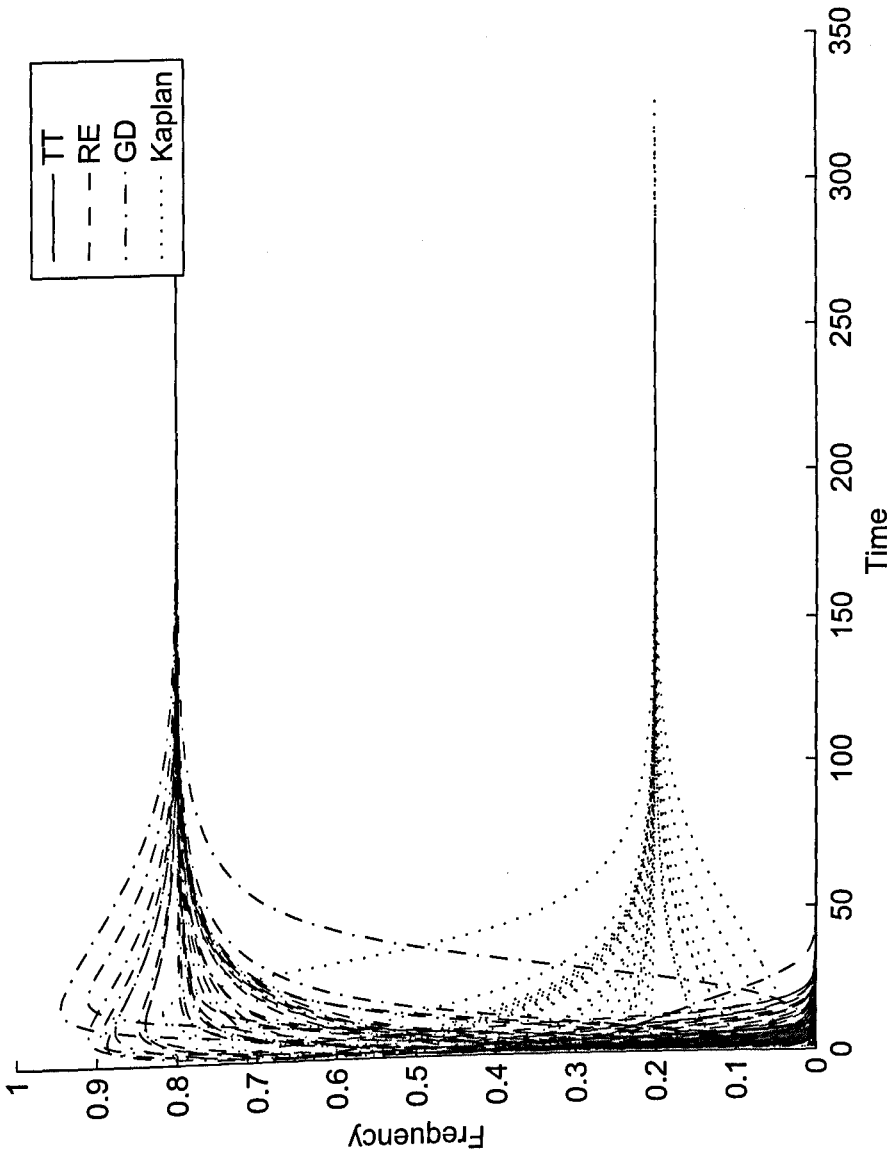

Figure 8.7: Replicator dynamics as time-series for CDA with 12 agents

can think of the proportion of the population m_i playing each pure strategy i as the *probability* of playing that pure strategy. Thus the vector $\vec{m}$ can be thought of as a mixed-strategy.

We can assess the relative likelihood of one strategy being adopted in long-term play relative to another by comparing the size of their respective basins of attraction. Figure 8.1 shows trajectories generated from $|M| = 250$ randomly sampled initial $\vec{m}$ vectors. For now, we assume that every initial mixed-strategy is equally likely to be adopted as a starting-point for the co-evolutionary process, and so we randomly sample the initial values of $\vec{m}$ from a uniform distribution and plot their trajectories as they evolve according to equation 7.1.

For four strategies, the direction-field is slightly trickier to display. Figures 8.3 to 8.7 show the convergence to equilibrium as a time series, when we consider all four strategies in both the CDA and the CH for $|A| = 4$, $|A| = 6$, and $|A| = 12$ agents.

To automate the analysis of institutions, we need to be able to provide some metric that allows us to quantify their performance in this kind of analysis. Different equilibria will yield different outcomes and different values of our design objectives, such as market efficiency, and we would like to weight these according to their likely-hood. In other words, we would like to compute the size of the basin of attraction of each equilibrium, in order to arrive at a probability of the equilibria actually occurring, and use this to calculate the expected value of our design metrics.

Table 8.2 shows the values of β (equation 8.2) for those $\vec{x}$ for which $\beta(\vec{x}, M) > 0$. These were obtained by taking a random sample M of size $|M| = 10^3$, and solving the replicator dynamics equation numerically. Stationary points that occur with a probability less than 10^{-2} were eliminated from the analysis as an approximation method to test for Lyapunov stability. Thus I take the stationary points reported in table 8.2 as *equilibrium solutions* and the value of β as the probability of arriving at the reported equilibrium. So for example in the top-left cell of table 8.2 we see that in a CH with $|A| = 4$ agents there are two pure-strategy equilibria: (i) at coordinate $(0, 0, 1, 0)$ in the simplex representing pure GD; and (ii) at coordinate $(0, 0, 0, 1)$ representing pure TK. The first equilibrium has a probability 0.39 of being played whereas the latter has a probability 0.61.

The value of U in each cell of table 8.2 denotes the pure strategy payoffs obtained in each particular experiment; that is, the heuristic-strategy payoff obtained to each pure strategy when all agents adopt it. So for example in the bottom-right cell we see that in a CDA with $|A| = 12$ agents we obtain payoffs $(0.85, 0.89, 0.99, 0.90)$ to strategies TT, RE, GD and TK respectively under homogenous adoption.

Similarly for the other cells in the table.

8.5 Discussion

With probabilities over outcomes, we are now in a position to assess the design of each mechanism. The value of EA in each cell of table 8.2 shows the expected efficiency of the mechanism. This is computed by weighting the pure-strategy payoffs U according to the probability of the pure strategy being played. For example, in the case of CDA with $|A| = 6$ agents, we see that there are two possible equilibria. The first equilibrium,

CH

$\|A\| = 4$	$\|A\| = 6$	$\|A\| = 12$
$\beta(0,0,1,0) = 0.39$ $\beta(0,0,0,1) = 0.61$	$\beta(0,0,1,0) = 0.31$ $\beta(0,0,0,1) = 0.69$	$\beta(0,0,0,1) = 1$
$U = (1.00, 0.90, 1.00, 1.00)$ $EA = 1.00$	$U = (1.00, 0.92, 1.00, 1.00)$ $EA = 1.00$	$U = (1.00, 0.93, 1.00, 1.00)$ $EA = 1.00$

CDA

$\|A\| = 4$	$\|A\| = 6$	$\|A\| = 12$
$\beta(0,0,0,1) = 1$	$\beta(0,0,0.84,0.16) = 0.97$ $\beta(0,0,0,1) = 0.03$	$\beta(0,0,0.8,0.2) = 1$
$U = (0.89, 0.86, 0.98, 0.89)$ $EA = 0.89$	$U = (0.85, 0.88, 0.98, 0.86)$ $EA = 0.96$	$U = (0.85, 0.89, 0.99, 0.90)$ $EA = 0.97$

Table 8.2: Heuristic-strategy equilibria over (TT, RE, GD, TK) for CH versus CDA

$(0,0,0.84,0.16)$, has a probability of 0.97 of being adopted. In this equilibrium the strategy GD has a probability 0.84 whereas the strategy TK has a probability of 0.16. By examining the payoffs to each of these strategies we can compute the expected efficiency of the mechanism in this equilibrium: $0.84 \times 0.98 + 0.16 \times 0.86 = 0.96$. In the second equilibrium we see that the strategy TK has a probability 1 of being played, hence the efficiency of this second equilibrium is 0.86. We then weight our overall efficiency according to the probability of each equilibrium: $0.96 \times 0.97 + 0.86 \times 0.03 = 0.96$.

First of all, since there non-truthful equilibria in all experiments we can conclude that TT is not dominant, and hence neither the CH or CDA mechanism is strategy-proof in these scenarios.

As expected from our discussion in Section 3.4, we observe that payoffs under truthful bidding in a CDA are relatively low: $EA = 0.85$ for $|A| = 6$ and $|A| = 12$. This might suggest that the CDA itself has a rather low efficiency. However, in order to assess the efficiency of the CDA we must take into account the fact that in these scenarios truth-telling is dominated. In fact, we see that various mixtures of GD and TK are likely outcomes, yielding efficiencies of between 0.89 and 0.97.

Thus although the CDA yields lower surplus, it is not as inefficient as we might expect had we assumed that it was designed according to incentive-compatibility criteria. As [50] points out, the main reason for choosing a CDA rather than a CH is to handle larger volumes of trade, and our results here suggest that this is a reasonable trade-off. Switching to a CDA from a CH as the New York Stock Exchange did in the late 1860s [12, p. 29], does not seem likely to entail a large loss of efficiency when we have relatively few ($|A| = 6$ or $|A| = 12$) traders in the market.

For the most part efficiency outcomes are deterministic - there is either a unique equilibrium that captures the entire simplex or all equilibria yield the same efficiency. The exception is the CDA with $|A| = 6$ agents. Here we have a mixed TK and GD equilibrium with efficiency $EA = 0.97$ versus a pure TK equilibrium with a *significantly lower* efficiency of $EA = 0.86$. Since the TK equilibrium has a very small basin of attraction $\beta = 0.03$ we conclude that the lower efficiency outcome is not very likely,

and hence if we have no prior knowledge of existing strategy frequencies in the trading population at large we assume a uniform distribution over starting points $M \subset \Delta$ and conclude that our efficiency is still likely to be very high. However, in the case where we do have prior knowledge about the frequency of strategies, e.g. we are tasked with evaluating a proposed choice of a continuous-clearing rule for a six-agent marketplace in which we *already* observe high proportion of sniping, then we might conclude that the pure TK equilibrium is much more likely to be reached (since we will be starting within its attractor), and thus we might recommend that CH clearing is used instead in order to avoid the probable efficiency hit predicted by our analysis. This hypothetical design tweak would yield an efficiency gain of $0.97 - 0.86$, or 11 percentage points, at the expense of transaction throughput. Thus by analysing the strategic *dynamics* of a proposed mechanism, we can perform *evolutionary* mechanism design whereby we make design decisions under *legacy* constraints (in this hypothetical scenario our legacy constraint is an existing marketplace populated by snipers). Evolutionary mechanism design is analogous to evolutionary game theory in that just as players may be constrained to gradually adjust their strategies, similarly mechanisms cannot always make instantaneous adjustments in their rules irrespective of what strategies are currently in play. We shall return to this discussion in Section 10.2.

In Chapter 1, I introduced the double-auction as an example of a self-organized complex system (SOCS). With the dynamic analysis in the previous sections, we begin to see what this means. In a traditional mechanism design scenario we simply demonstrate that under our proposed mechanism truth-telling is dominant and that efficiency under truth-telling is maximised. We then assume that truthful behaviour will be instantly adopted and that our mechanism will remain forever efficient in stasis. In contrast, the picture I paint here is a dynamic and uncertain one. Real-life considerations and multiple design objectives mean that we can rarely demonstrate that a simple, prescribed strategy such as truth-telling is dominant. Rather, we have multiple equilibria within a dynamic system comprised of discrete non-linear components, and we are not always certain how the ensemble will evolve. As with other complex systems, it is extremely difficult to discover the system's likely behaviour using analytical methods. Using computationally-intensive numerically methods such as the empirical game-theoretic analysis conducted in this chapter we can get an *insight* into the dynamics of the system and make some tentative forecasts.

However, as with other complex systems, such as meteorological ones, we should take forecasts of them with a pinch of a salt, especially in the long term. For example, one of the potential drawbacks of our analysis is that we have only considered a small subset of the space of possible strategies, and one of these, the RE strategy, has many internal parameters. Is it not conceivable that if a new strategy (for example, a variant of RE with tweaked parameters) were introduced into our market ecosystem that it would upset the equilibria that we have so carefully analyzed and cultivated? We shall address this question in detail in the next chapter, but the brief answer is: yes. As with other engineering design methodologies [10, 6], real-life mechanism design is an *iterative process*; we do the best that we can to analyze anticipated outcomes, but a complete and future-proof analysis is wholly intractible, and thus at some point reality will inevitably overtake our initial predictions and we will have to adjust our design in light of up-to-date empirical observations of the system *in vivo*. As discussed in the

previous paragraph, the methods introduced here will allow us to do just that. Thus rather than simply launching a theoretically optimal auction design onto the world, instead, as *evolutionary* mechanism designers we design, analyse, observe, tweak and then repeat.

One topic that has received considerable interest within economics over recent decades is that of viewing markets as a particular class of SOCS that exhibits a property called self-organized criticality (SOC) [4], meaning that the attractors of the system lie on critical points (eg. phase transitions) between order and chaos. These critical points exhibit sufficiently dynamical behavior that the system does not "freeze" into low complexity configurations, but at the same time their dynamics is sufficiently ordered that the system does not "boil" into noise, and hence this regime is highly conducive to complexity. Since these systems naturally have attractors located at critical points, they tend to be continuously "poised" between order and chaos, and hence they naturally equilibrate towards complex states. The existence of very simple physical systems that possess critical-point attractors strongly suggests that self-organised criticality may be responsible for much of the complexity that we observe in natural systems. One of the characteristics of critically-poised systems is scale-invariance hence their macroscopic properties tend to follow power-law relationships. It is suggested that the long-tail distribution of time intervals between events such as market crashes in the business cycle are due to markets being critically-poised in this manner [81, 88].

In contrast, our analysis yields more well-ordered, non-chaotic dynamics. If we do indeed observe power laws and chaotic behaviour as a result of criticality in real markets[6], this raises two questions:

- Are these methods applicable under chaotic dynamics (and hence to real markets)?
- Has the behaviour of the system been oversimplified?

The answer to both questions is yes. As regards the former, if the replicator-dynamics *had* yielded chaotic dynamics for the underlying heuristic-game (as it can do for certain payoff structures [130]), we could have still computed basin sizes for the resulting strange attractors and computed expected values of our design objectives. As regards the latter, *any* analysis has to abstract and simplify in order to be useful; in this analysis we are taking a very high-level view of the system in order to assess its macroscopic design properties. If we were to look under the hood, and plot the evolution of the state variables comprising each agent's strategy (which are still being computed and accounted for by the underlying heuristic-strategy analysis), we would likely see more entropy in the underlying system. For example, the RE strategy chooses its actions probabilistically in contrast to the deterministic evolution of the replicator-dynamics equation, and it is not inconceivable that we would observe criticality if we were to examine actual bid prices as a time-series at this level.

However, ultimately, at the macroscopic-level of the system our analysis is based on the replicator dynamics. Although as discussed the replicator-dynamics can exhibit

[6] In fact, this is highly contentious, as is the question of whether criticality is actually observed in real-life sand-piles [72, p. 14] as opposed to the simulated sand-piles in Bak et al.'s original SOC paper [4].

chaos, it does not belong to the class of systems which typically exhibit criticality. The replicator dynamics was originally introduced by Maynard-Smith [86] to model biological evolution in terms of gradual adjustment to equilibrium, as originally envisaged by Darwin [31]. However, Eldredge and Gould [40] argued that such "gradualist" models were oversimplistic, and put forward an alternative theory of evolution based on the concept of "punctuated equilibrium", which is closer to the view of self-organized criticality.

The debate over gradualism versus punctuated equilibrium has never been settled and rages on [133]. However, in future work I will use alternative dynamic models of learning and evolution, such as those discussed by Jensen [72, p. 73], and conduct a sensitivity-analysis similar to that described in the next chapter in order to assess whether these forecasts are sensitive to alternative models of strategy adjustment.

8.6 Summary and Contribution

Recall from section 1.3 that our method for evolutionary mechanism design is outlined as follows:

input : A set of initial heuristic strategies S, and a legacy mechanism μ
repeat
  $S \leftarrow$ FiSH+ (S, μ);
  publicise S to participants;
  $\vec{x} \leftarrow$ *frequency of each strategy observed in vivo*;
  $S \leftarrow S \cup \{$ *strategies observed in vivo* $\}$;
  $\Lambda \leftarrow$ *space of feasible variants of* μ;
  $\mu \leftarrow \arg\max_{\mu* \in \Lambda}$ EvaluateDesignObjectives$(\mu*, S, \vec{x})$;
  implement rules defined by μ;
until forever;

In this chapter I have described how to evaluate the function:

$$\text{EvaluateDesignObjectives}(\mu, S, \vec{x})$$

where μ denotes a mechanism, S denotes a set of heuristic strategies and $\vec{x} \in \Delta^{|S|}$ denotes a weighting over strategies based on current observations of the frequency with which each strategy is in play *in vivo*. I have shown empirically that this yields useful results for $S = \{\text{TT}, \text{RE}, \text{GD}, \text{TK}\}$ for each of the mechanisms $\mu =$ CH and $\mu =$ CDA. I also demonstrated that our design objectives can be sensitive to $\vec{x}$: in the case of $\mu =$ CDA and $|A| = 6$, if we observe $\vec{x} = (\delta, \delta, \delta, 1 - \delta)$ where δ is small, that is a situation in which a high proportion of traders using the sniping strategy TK in the real-life mechanism, then our assessment of our design objectives will be different to that when we assume a uniform weighting $\vec{x} = (\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4})$, and thus I have demonstrated that this method can take into account legacy considerations.

One of the potential criticisms of this kind of analysis is that it is highly sensitive to the set of heuristic strategies S, which can never be truly comprehensive for an initial

design. The next chapter will explicitly deal with this criticism by setting the mechanism design problem in the iterative context implied by the above pseudo-code, and we shall discuss the `FiSH` algorithm for refining our initial heuristic-strategy analysis by searching for hitherto unanalysed strategies that might break our existing equilibria.

Chapter 9

Searching the Space of Strategies

In the previous chapter, we used a heuristic-strategy analysis to analyse two variants of the double auction market mechanism populated with a mix of heuristic strategies, and were able to find approximate game-theoretic equilibrium solutions. In this chapter, we shall use the same basic framework, but focus on the CH mechanism with uniform pricing (Section 3.3.1). Our goal will be to use ideas from empirical game-theory in order to search the space of trading *strategies*, whilst restricting attention to a single mechanism.

Initially we will start with a subset of three heuristic strategies from the original set of four discussed in the previous chapter: TT, RE and GD, which are summarised in Table 9.1.

As in the previous chapter (Section 8.4), since all mixed-strategy vectors lie in the unit-simplex, for $k = 3$ strategies we can project the unit-simplex onto a two dimensional space and then plot the switching between strategies that occurs under the dynamics of equation 7.2. Figure 9.1 shows the direction-field of the replicator-dynamics equation for these three heuristic strategies, showing that we have two equilibrium solutions. Firstly, we see that GD is a best-response to itself, and hence is a pure-strategy equilibrium. We also see it 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. Additionally, there is a second mixed-strategy equilibria at the coordinates (0.88, 0.12, 0) in the simplex corresponding to an 88% mix of TT

Abbreviation	*Description*
TT	The truth-telling strategy, (section 4.1.1)
RE	The reinforcement-learning strategy (section 4.2.4), configured with the Roth-Erev learning algorithm (section 4.2.4)
GD	The Gerstad-Dickhaut strategy (section 4.2.3)

Table 9.1: The initial heuristic strategies chosen for the analysis

and a 12% mix of RE. However, the attractor for this equilibrium is much smaller than the pure-strategy GD equilibrium; only 6% of random starts terminate here vs 94% for pure GD. Hence, according to this analysis, we expect most of the population of traders to adopt the GD strategy.

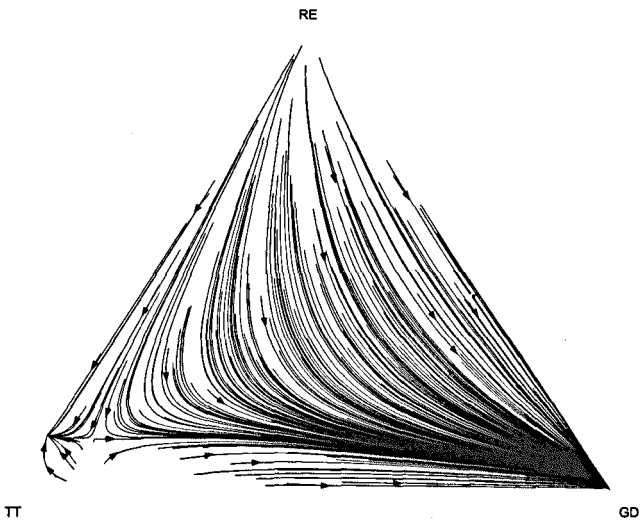

Figure 9.1: The original replicator dynamics direction field for a 12-agent clearing-house auction with the original unoptimized Roth-Erev strategy (labeled RE).

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 9.2 shows the direction-field plot after we perform a perturbation where we remove 2.5% of the payoffs from the TT and GD strategies and assign +5% payoffs to the RE strategy. This results in a qualitatively different set of equilibria; the RE strategy becomes a best-response to itself with a large basin of attraction (61%), and thus we conclude that our equilibrium analysis is sensitive to small errors in payoff estimates, and that our original prediction of widespread adoption of GD may not occur if we have underestimated the payoffs to RE.

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 points to* RE *as a candidate for potential optimization.*

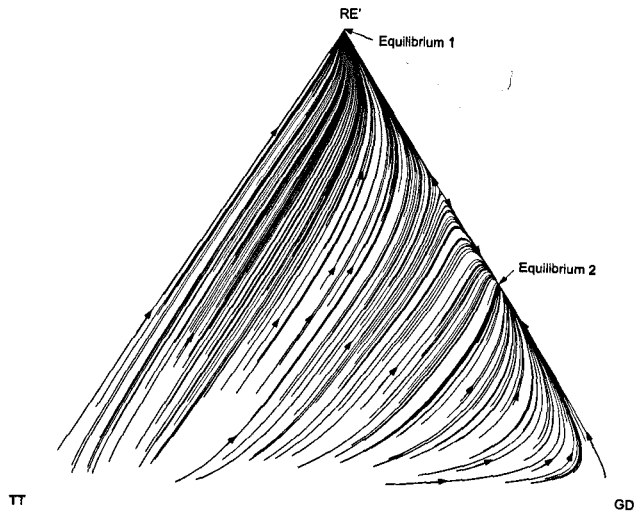

Figure 9.2: Replicator dynamics direction field for a 12-agent clearing-house auction perturbed with +5% payoffs to the Roth-Erev strategy (labeled RE')

9.1 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.

9.1.1 Strategy space

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 4.2.4. 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 4.5 to 4.8.

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 $(0, 1)$.

9.1.2 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. Basin size was estimated using the same brute-force methods described in Section 8.4, but since I recompute all entries in the heuristic-payoff matrix in support of each candidate strategy, I 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. I 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 fitness function is derived from equation 8.2:

$$F(i, S, [H]) = \sum_{\vec{x} \in \epsilon_{[H]S}} \beta_{[H]}(\vec{x}, M) x_i \tag{9.1}$$

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 8.2 (p. 90), 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:

$$S = \{\text{s*}, \text{TT}, \text{GD}, \text{RE}\}$$
$$i = 1$$

where s* is our candidate strategy. 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 on p. 109; I call this the `FiSH` algorithm, since we will use it to "fish" for a new heuristic strategy.

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

input : A set of heuristic strategies $S = \{s_1, s_2, \ldots s_n\}$
output: A new heuristic strategy OS

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

Algorithm 2: FiSH

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.

9.2 Results

Figure 9.3 shows the mean fitness of the GA population for each generation. As can be seen, there is still a large amount of variance in fitness values in later generations. 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:

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

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

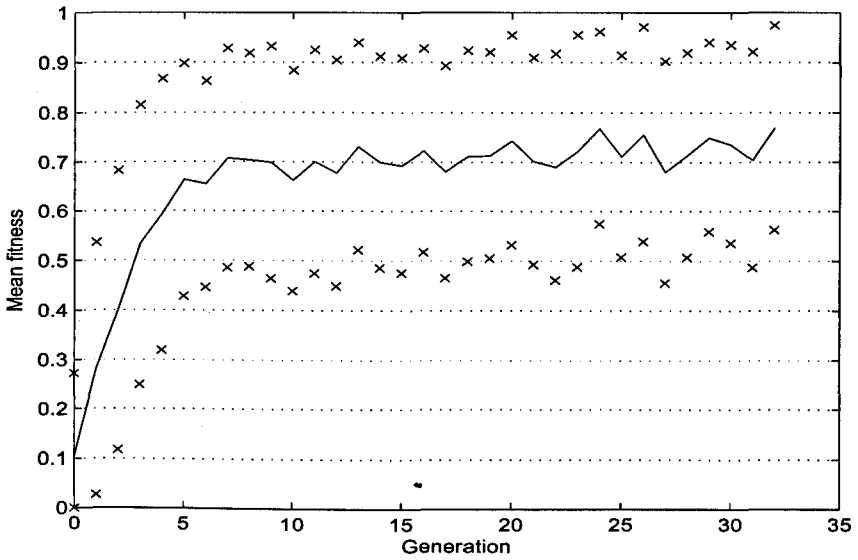

Figure 9.3: Mean fitness of the GA population with one standard deviation

strategies in the last ten generations, and is another factor that indicated convergence of the search.

I 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. With the current version of my simulator[1], I am able to complete this analysis in less than twenty four hours using a dual-processor 3.6Ghz Xeon workstation.

Figure 9.4 shows twenty trajectories of the replicator-dynamics plotted as a time-series 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, I 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 (OS, TT, GD, 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:

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

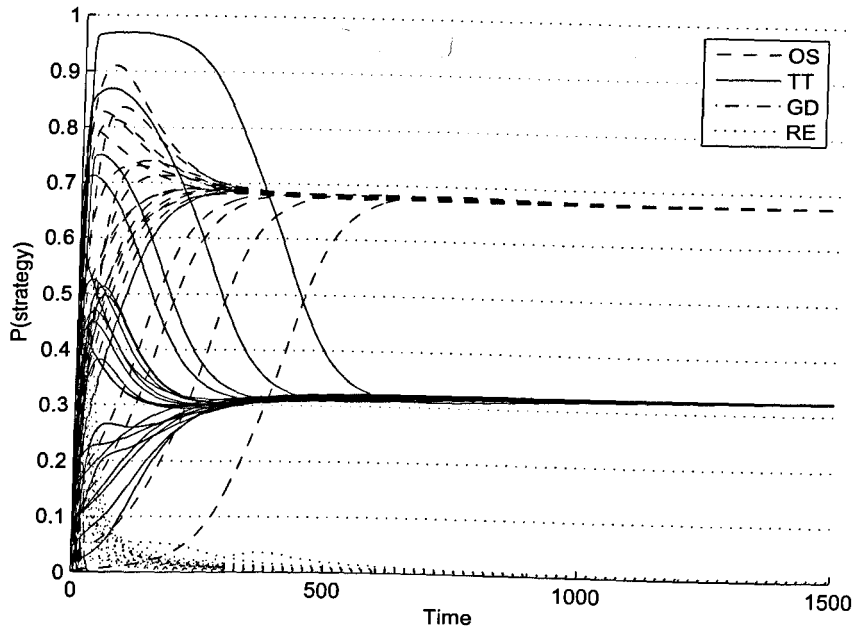

Figure 9.4: Replicator dynamics time series plot for a 12-agent clearing-house auction showing interaction between optimised strategy (OS) versus GD, TT and the original Roth-Erev strategy (RE)

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

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 9.5 and 9.6 show the direction field for two of the 3-strategy combinations involving our optimised strategy: (OS, TT, GD) and (OS, GD, RE) respectively.

9.3 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?

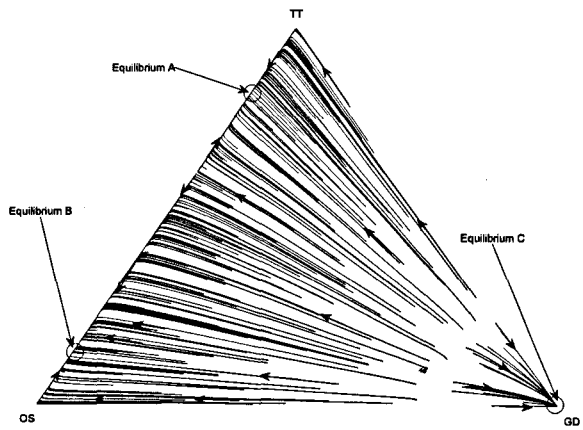

Figure 9.5: Replicator dynamics direction field for a 12-agent clearing-house auction showing interaction between optimised strategy (OS) verses TT and GD

As discussed earlier, we use the same method of assigning valuations as in [145]; 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:

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

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:

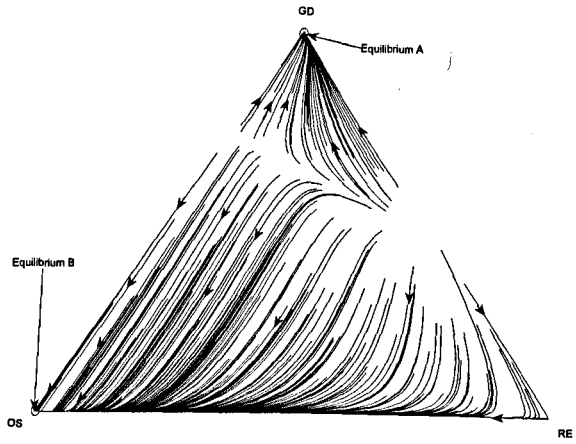

Figure 9.6: Replicator dynamics direction field for a 12-agent clearing-house auction showing interaction between optimised strategy (OS) verses GD and the original Roth-Erev strategy (RE)

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

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 it is far away from the equilibrium-price it can adjust its margin close to zero, whilst if it is near the equilibrium-price it 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 I will examine this hypothesis in more detail and conduct a statistical analysis in which I 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, I 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.

9.3.1 An iterative approach

In this chapter, 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, and the algorithm, called FiSH, is illustrated on p. 109.

We have validated FiSH empirically by applying it to a highly complex game, the double-auction, and demonstrated that it is capable[2] 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 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 FiSH+ (p. 115) illustrates this proposed algorithm.

9.3.2 Strengths and Weaknesses

The strength of this method for strategy acquisition is its ability to be applied in realistically complex games (mechanisms). However, just as the domain to which I 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 FiSH combines two fields in a very simple way, each with a growing analytical literature, viz. empirical game-theory and optimisation. Additionally, I have demonstrated that the algorithm works effectively in at least one highly complex setting, thus we have an existence proof that the algorithm is effective in at least one realistically complex scenario. For the empirical study in this chapter I have used a general purpose optimisation method, that is a genetic algorithm. However, future work 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. [146].

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

I have not attempted to validate the proposed iterative version of the algorithm, FiSH+, in this thesis. 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 analaysis of how the algorithm might converge prior to investing in a full empirical case study.

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
$[H] \leftarrow$ GetHeuristicPayoffMatrix(S, μ);
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**
      $\hat{F} \leftarrow F(i, S, [H]')$;
      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* $\hat{\text{OS}}$;
  OS $\leftarrow$
  $\arg\max_{\text{s*}\in\Pi} F(1, \text{s*} \cup S, \text{GetHeuristicPayoffMatrix}(\text{s*} \cup S, \mu))$;
  $S \leftarrow \text{OS} \cup S$;
  $[H] \leftarrow$ GetHeuristicPayoffMatrix(S, μ);
  $S \leftarrow$ *eliminate dominated strategies from* S *based on* $[H]$;
until forever ;

Algorithm 3: FiSH+

9.4 Summary and Contribution

In the previous chapter we performed a quantitative analysis of the design properties of two different auction mechanisms using an initial set of four heuristic-strategies. We also asked the question as to how stable our analysis was given that we had only chosen a small set of strategies, one of which had many free parameters. By applying the FiSH+ algorithm we can answer this question; we can see if there are additional, previously unconsidered strategies that break our initial equilibrium. In this Chapter we have refined an initial analysis of the CH based on the three strategies TT, RE and GD, and discovered a new incumbent strategy OS with large attractors (basin size), which

are stable under payoff perturbations. To a mechanism designer, this latter state of affairs is particularly attractive, *since larger, more stable basin sizes correspond to more deterministic, and hence predictable behaviour.* In a legacy mechanism design scenario, if we are able to provide an equilibrium analysis over existing strategies which demonstrates similarly clear-cut equilibria, then we may be able to convince participants that these are the best-response strategies that their competitors are likely to adopt, and therefore that they should adopt also. If we then make the algorithms corresponding to our *new* heuristic strategies freely available to participants, and if they believe our equilibrium analysis, then they are likely to play our prescribed strategies, thus bringing about our predictions, and hence maximising our design objectives. By finding new strategies with large stable attractors, we make our equilibrium analysis more believable to participants. This is analogous to incentive-compatibility in a conventional mechanism design scenario, where it is clear to participants that TT is the traders' best-response to the mechanism: in an incentive-compatible mechanism TT is a "freely-available" strategy with a large attractor. In realistically complex mechanisms such as the double-auction, TT is dominated. However by applying the FISH+ algorithm we can find analogs of TT for complex mechanisms.

Of course, in our new equilibria, our existing mechanism rules may no longer maximise our design objectives. In the previous chapter, we described real-life mechanism design as an iterative process (section 8.5), and that is exactly how evolutionary mechanism design addresses this issue. Thus our algorithm for evolutionary mechanism design is as follows:

input : A set of initial heuristic strategies S, and a legacy mechanism μ
1 **repeat**
2 | $S \leftarrow$ `FiSH+` (S, μ) ;
3 | *publicise S to participants*;
4 | $\vec{x} \leftarrow$ *frequency of each strategy observed in vivo*;
5 | $S \leftarrow S \cup \{$ *strategies observed in vivo* $\}$;
6 | $\Lambda \leftarrow$ *space of feasible variants of* μ;
7 | $\mu \leftarrow \arg\max_{\mu* \in \Lambda}$ EvaluateDesignObjectives$(\mu*, S, \vec{x})$;
8 | *implement rules defined by* μ;
9 **until forever** ;

In this chapter I have demonstrated how could step 2 can be automated. In the previous chapter, we saw how to semi-automatically compute the function in step 7. In the next chapter I shall describe how step 7 can be fully automated.

Chapter 10

Searching the Space of Mechanisms

Recall that our method for evolutionary mechanism design is as follows:

input : A set of initial heuristic strategies S, and a legacy mechanism μ
1 **repeat**
2 $S \leftarrow$ FiSH+ (S, μ) ;
3 *publicise S to participants*;
4 $\vec{x} \leftarrow$ *frequency of each strategy observed in vivo*;
5 $S \leftarrow S \cup \{$ *strategies observed in vivo* $\}$;
6 $\Lambda \leftarrow$ *space of feasible variants of* μ;
7 $\mu \leftarrow \arg\max_{\mu* \in \Lambda}$ EvaluateDesignObjectives$(\mu*, S, \vec{x})$;
8 *implement rules defined by* μ;
9 **until forever** ;

In chapters 8 and 9 we examined methods for evaluating design objectives and iteratively searching for new heuristic strategies (FiSH+) respectively. In this chapter we shall turn attention to step 7, that is, the problem of searching the space of mechanism rules in order to solve the optimisation problem:

$$\arg\max_{\mu* \in \Lambda} \text{EvaluateDesignObjectives}(\mu*, S, \vec{x})$$

Rather than consider the entire space of possible mechanisms, we shall take as Λ the space of possible transaction pricing rules for a CH mechanism, the different forms of which are discussed in sections 3.3.1 to 3.3.3. Recall that there are two main variants of pricing rules for this institution: uniform pricing in which we set the transaction price based on the market quote $(e\hat{q}_a, e\hat{q}_b)$, and discriminatory pricing in which we set the transaction price based on the individual bid and ask prices. Each of these rules is parameterised by a constant $k \in [0, 1]$ which determines where we will set the

transaction price in either the interval between $e\hat{q}_a$ or $e\hat{q}_b$, or the bid and ask prices depending on whether we are using uniform or discriminatory pricing respectively. In a $k = \frac{1}{2}$ mechanism we set the price halfway between the two relevant values. In a $k = 1$ mechanism we set transaction prices at the bid price, or the bid component of the market quote. Similarly in a $k = 0$ mechanism we set prices at the ask price, or the ask component of the market quote. For extreme values values of k there are clear-cut analytic incentive-compatibility results for buyers ($k = 0$) or sellers ($k = 1$). However, there is no clear-cut analysis of how we should choose k in the general case.

In the following section I briefly review earlier work in which I attempted to use a co-evolutionary algorithm to solve the optimisation problem. In section 10.2 I discuss the relationship between co-evolutionary algorithms and game-theory in order to demonstrate why this earlier approach is not suitable for the evolutionary mechanism design algorithm outlined at the beginning of this thesis. In section 10.3 I outline the non-coevolutionary optimisation approach that I adopted in order to circumvent these problems. In section 10.4 I describe an experiment to empirically validate this optimisation approach, the results of which can be found in section 10.5.2. Finally I conclude with a discussion and summary.

10.1 A review of earlier work

As discussed in sections 3.3.1 to 3.3.3, the transaction pricing rule sets the price of any given transaction as a function of the *bid* and *ask* prices submitted by buyers and sellers respectively. In a private-values trading scenario, bids and asks can be thought of as signals [39, p. 395] from the traders expressing their valuation for the resource being traded. The difficulty the auctioneer faces in allocating the resource to those who value it most highly (i.e. achieving an optimal allocation or maximum market efficiency) is that these signals cannot necessarily be relied upon to be truthful; agents might misreport their valuations in order to make profit at the expense of others. One technique to counter this problem is to design *incentive-compatible* mechanisms which have the property that the best strategy for every agent is to report their valuation truthfully. This is typically achieved by forcing agents to back up their value claims with hard cash, thus imposing a "handicap" on the signals of the traders, and encouraging honest signalling through the handicap principle [162]. Successful application of this principle involves careful reasoning about how to set the handicap, i.e. the transaction price, as a function of the signal, i.e. the bids and asks of the traders.

As discussed in Section 3.3.4, the uniform-price CH can be shown to be incentive-compatible for sellers for $k = 1$, and incentive-compatible for buyers for $k = 0$. However, there is no value of k for which the mechanism is incentive-compatible for all traders.

In earlier work [107, 108], two possible approaches were used to analyse the space of possible transaction pricing rules using computational techniques, with a view to finding rules which optimise various design objectives.

In the first approach, co-evolutionary machine learning was used to simulate an evolutionary "arms-race" between populations of trading strategies and a separate population of pricing rules (the mechanism population) [107]. Individuals in each popula-

tion were represented as lisp expressions and evolved using Koza genetic-programming [79]. The fitness function for the strategy populations was a function of the individual profits of traders playing those strategies, and the fitness function for the pricing rule population was a function of the overall market efficiency achieved by an auctioneer using that rule against the current strategy populations.

In these early co-evolutionary experiments, it was hoped that, as the strategy populations evolved predatory non-truthful strategies, the pricing rule population would respond by evolving defenses, and that over time incentive-compatible mechanism rules would evolve that were robust against a wide variety of trading strategies, in much the same way that prey populations adapt robust defenses against predator populations in co-evolutionary arms races in nature [35, 141]. Despite some promising preliminary results, it was found that this approach suffered from a number of drawbacks, mainly:

1. The co-evolving system rapidly descended into suboptimal auction mechanisms if the mechanism population was not artificially seeded with individuals with a minimum-level of initial fitness. In cases where the mechanism population started from extremely low fitness individuals, such as pricing rules which set the transaction price at 0 regardless of the signals arriving from traders, the strategy populations would try and fit to these artificially low-fitness mechanisms and evolve to a state where their bids were meaningless. Meanwhile the mechanism population would be unable to discover more rational rules which worked with the existing "broken" trading strategies. Therefore the trading strategies could not evolve to work with more rational mechanisms and so on.

2. Where more promising results were obtained by artificially seeding the mechanism population with initial promising rules, the results were highly ambiguous. Often the mechanism population would oscillate between stable states representing rules corresponding to $k = 0$ and $k = 1$. Initially speculation was that this was a reflection of the fact that there were no incentive-compatible rules for both buyers and sellers. The theory was that the mechanism population was settling on incentive-compatible rules for sellers, the buyer population was then responding by evolving non-truthful strategies, the mechanism population was responding by evolving rules that were incentive-compatible for buyers, the sellers were responding by evolving non-truthful strategies, ad infinitum. However, this theory proved unfounded as the explanation turned out to be that rules of the form $k = 0$ and $k = 1$ were more dense in the search space.

In the next section I take a game-theoretic perspective and attempt to explain why the co-evolutionary algorithm failed to produce meaningful results in this context.

10.2 Mechanism design as strategic-interaction

It is often instructive to analyse co-evolutionary processes in game-theoretic terms, since in a co-evolutionary interaction the fitness assigned to any given individual depends on the joint actions of the other individuals with which it interacts in a very

similar manner to an evolutionary game[1]. When we co-evolve auction mechanisms and trading strategies we are implicitly defining a game between two players[2] : the *mechanism player* on the one hand, and the *trader player* on the other. Each player attempts to maximise their payoff (analogous to maximising fitness); in our present scenario the mechanism player attempts to maximise market efficiency EA, whereas the trader player attempts to maximise utility u_i. Note that if the selection function of our co-evolutionary algorithm picks individuals from each population based on a stochastic function of fitness rather than phenotype, then we are implicitly modelling a game of *imperfect information*; the individuals in the population do not "know in advance" the action that is being adopted by any other. This has important implications which will be discussed further in the next section.

Ideally we would like to find the *optimal* strategy for the mechanism player. In game theory the concept of an optimal strategy is defined formally as a *dominant strategy*. In this chapter we will be restricting attention to the clearing rule (Section 3.2.9), so a hypothetical dominant strategy for the mechanism player would be a clearing rule that obtained a better payoff, EA than any other clearing rule, *no matter what strategy is adopted by the trader player*. However, not every game possesses a dominant strategy solution (and it is not apriori clear that we should expect the mechanism versus trader game to possess one). More commonly the concept of optimality in a game is *relative*; if a dominant-strategy does not exist then the best strategy to play depends on the strategy adopted by one's opponent(s).

Although not every game possesses a dominant-strategy, we know that *all* games possess at least one Nash *equilibrium* in which the strategy adopted by every player is a best-response to every other player's strategy. Consider a hypothetical equilibrium for our game at hand in which the mechanism population chooses a clearing-rule which sets the transaction price at a *fixed* constant value $\forall_i \ \text{price}(c_i) = d$ which is independent of the trader shout price, and in response the trader player adopts a strategy of always submitting shouts with zero prices: $\forall_i \forall_t \ \zeta(i,t) = 0$. Depending on the distribution of trader valuations, a rule which sets transaction prices close to the expected equilibrium price $d \approx E(p*)$ would achieve a reasonable expected payoff $E(EA) \approx 1$ for the mechanism player. From an external mechanism designer's point of view this clearing rule is clearly brittle and undesirable, especially if the variance in valuations and hence in efficiency is large. However, this hypothetical situation would be very hard to leave once we arrive at it, since if the mechanism player attempts to switch to conventional clearing rules which set transaction prices as a function of shout prices, it will be faced with the issue that all shout prices are 0. Similarly, the trader player cannot improve their payoff by unilaterally switching to any other strategy since their payoff is no longer a function of their shout price. This situation is a game-theoretic

[1]Note that this applies regardless of whether we intuitively think of our original problem as a game. Game theory is simply a mathematical tool that allows us to study co-dependent optimization problems- that is, what potential solution should we choose given that our choice will influence the solution of other optimizers and vice versa. This is precisely the scenario instantiated by a co-evolutionary algorithm, hence game-theory is an invaluable theoretical tool in understanding the properties of co-evolutionary systems.

[2]For conciseness and simplicity, in this section only we shall assume that many trading agents are under control of the single notional trader player, and that all the agents adopt the same strategy that is specified by the trader player at any given time. Note, however, that we will be dropping this simplifying assumption in the remainder of the chapter.

equilibrium of the mechanism versus trader game.

If equilibria such as these have large basins of attraction (p. 90) under the dynamics of our co-evolutionary process, then we should not be surprised if our co-evolutionary algorithm converges on them. Indeed, this was one of the major problems that was encountered in earlier work when I attempted to use co-evolution to evolve robust mechanisms: the co-evolutionary algorithm sometimes converged on what appeared to be game-theoretic equilibria, but it is not clear that the theoretical equilibrium solutions of the mechanism versus trader game are in any way desirable from a mechanism design perspective, as illustrated by the above example.

In co-evolutionary terminology equilibrium states such as Nash equilibria are sometimes referred to as "local optima" of the co-evolutionary algorithm. Note however that although they are referred to as "local" this does not imply the existence of a *global* optimum that still remains to be found, since as discussed above there is not necessarily a *dominant* strategy for the mechanism player. Additionally, it is not always the case that payoffs are straightforwardly maximised in these states relative to the majority of other points in the phase space of the system. For example: there may exist many alternative clearing rules which give better payoff EA to the mechanism player if the trader player were to adopt a different, non-equilibrium, strategy; or there may exist alternative equilibria strategy profiles which yield higher payoff to both players. Thus there may be multiple equilibrium points in the phase space of our co-evolutionary process. These are local "optima" in the sense that moving a very small distance away from them in the phase-space will not yield an increase in payoff (fitness). However it is important to note that there could be very many other "high-payoff" states further away in the phase space which yield a higher payoff to a given player than any any of our "locally optimal" strategy profiles. However these "high-payoff" states are not necessarily *Nash equilibrium* states.

Thus in general these "local optima" do not minimise (or maximise) any arithmetical function of their payoffs compared with non-equilibrium strategy profiles, and neither is their basin size under the co-evolutionary dynamics necessarily proportional to any of the payoffs. Hence if our co-evolutionary search converges on a Nash equilibrium, it is difficult to view this as a solution to a maximisation problem in which we are systematically searching for optimal, or even satisficing, mechanisms; indeed in our present scenario, in the absence of a dominant-strategy for the mechanism player it is not clear that the notion of an "optimal" mechanism has any meaning, since the optimal strategy for the mechanism player will be very sensitive to the strategy adopted by the trader player, and in the most likely case where there are multiple Nash equilibria ("local optima") for the game, there will be many possible "locally optimal" strategies that the trader player could adopt in the long term.

However, the Nash equilibria of the mechanism versus trader game are useful solutions to a different, but interesting, problem. If we are modelling a process in which multiple competing market institutions asynchronously adjust their rules over repeated interactions in response to observed trader strategies in the real world, and vice versa, (analogous to the scenario analysed by Roth and Ockenfels [120] in which they compare two competing online auction formats: eBay and Amazon), then we might expect equilibrium solutions such as the fixed-price clearing rule to be the rational end result. It is not inconceivable, for example, that the reason that we continue to see a prevalence

of fixed-price institutions such as bricks-and-mortar shops for selling consumer goods in real market places, despite the possibility of dynamically-priced institutions such as eBay[3], is due to fact that fixed-price institutions are an equilibrium solution of the real-life co-evolution between market mechanism and trader behavior. For example, consumers may be unable to switch from a fixed-price to an auction market for their required good since one may not exist yet, and correspondingly it may be very difficult for a startup to create an online auction market in the absence of existing traders on either side of the market. As well as accounting for historical and present observations of actual market behavior, this analysis could also be *normative*; we might *recommend* that retailers adopt a fixed-price mechanism based on the fact that it is a best-response to the likely status quo. In this case we might interpret our solution as "the optimal" one in some sense.

10.3 Mechanism design as optimization

In the previous section we saw that co-evolutionary algorithms are natural models of games of imperfect information, or simultaneous move games. The previous experiments could be thought of as an analysis of evolutionary mechanism design in the case that the mechanism designer and the traders are simultaneously attempting to anticipate the choice of the other.

However, our algorithm for evolutionary mechanism design is a *sequential* iterative process involving a single institution. In this case, the considerations from the previous section do not apply, since the mechanism designer is given the opportunity to move first by announcing their mechanism rules publicly to the trader population, who then respond by placing shouts in the mechanism. In this scenario we no longer have a repeated simultaneous-move game, instead we have a 2-move extensive-form game. In the first move the mechanism player announces their mechanism rules with *perfect information*, and in the second move the trader player responds by placing shouts. In contrast to the previous section, in this scenario the trader player does not have to attempt to "anticipate" the move made by the mechanism player; rather it can form its strategy conditionally based on the mechanism rules chosen by the mechanism player. Thus, as a mechanism designer we should choose the optimal mechanism rules in the sense that the chosen rules optimise our design objectives when the trader player plays their best strategy *under that particular chosen mechanism.*

This scenario is not straight-forwardly modelled by a standard co-evolutionary algorithm; rather it is more natural to view it as a non-co-evolutionary optimisation problem in which we evaluate each potential mechanism by computing the values of our design objectives when traders play their best strategy for our candidate mechanism. However, this problem is complicated by the fact that although the traders are not attempting to anticipate the mechanism rules (since these are already known), they are having to anticipate the moves of other traders (since there may be more than one trader, and they will be interacting under imperfect information).

Rather than attempting to compute the full Bayesian-Nash equilibria (which would be intractable) for the trading strategies, I have adopted an empirical game-theoretic

[3] `http://www.ebay.com/`

approach based on the RL strategy (Section 4.2.4) and the RE learning algorithm (Section 4.2.4). My rationale for choosing this combination is that it forms the basis of a *cognitive model* of how people actually behave in strategic environments. In particular it models two important principles of learning psychology:

- *Thorndike's law of effect*—choices that have led to good outcomes in the past are more likely to be repeated in the future; and
- *The power law of practice*—learning curves tend to be steep initially, and then flatten out.

The Roth-Erev algorithm belongs to a class of game-playing models known as "stimuli-response" models. These models have much in common with the replicator dynamics model of evolutionary game theory [86], and as in evolutionary game theory, the stable asymptotic behaviour of a multi-agent simulation using the Roth-Erev learning model can be interpreted similarly to the Nash-equilibrium of classical game theory or the evolutionary-stable-strategy of evolutionary game theory; stable states constitute strategy sets that are hard-to-leave and are likely to persist once they are reached, even when we consider agents who are not using the actual Roth-Erev learning algorithm to form their strategy. Hence, one way of viewing the analysis in this chapter is as an empirical game-theoretic analysis similar to that presented in Chapter 8, but in which the choice of heuristic strategies corresponds to each markup selected by the $RL_{\lambda_i}(t)RL_{\mu_i}$ term of equations 4.51 and 4.52. The principle advantage of this approach over a full heuristic-strategy analysis is the reduced computational overhead.

It is common to view mechanism design as the search for a mechanism that optimises a single parameter—market efficiency, for example. In contrast, in this chapter we shall consider mechanism design to be a *multi-objective optimization* problem in which we simultaneously maximise several parameters— market efficiency and trader market power being two we consider in this chapter. The difficulty in doing this lies in simultaneously maximising as many dimensions as possible.

In the remainder of this chapter, I describe how I have used these ideas to carry out some experiments in automated mechanism design in the setting of a deregulated electricity market.

10.4 Experimental setup

The experimental scenario stems from [98] (hereafter referred to as *NPT*), as described in detail in Section 6.3. To recap, a number of traders buy and sell electricity in a discriminatory-price[4] continuous double auction. Every trader assigns a value for the electricity that they trade; for buyers this is the price that they can obtain in a secondary retail market and for sellers this reflects the costs associated with generating the electricity. Here this value is considered *private*; because traders are always trying to make a profit themselves, sellers are not willing to reveal how little they might accept for units of electricity and buyers are not willing to reveal how much they might pay for

[4] In *uniform* price auctions, all trades in any given auction round happen at the same price. In *discriminatory* price auctions of the kind we have here, different trades in the same auction round occur at different prices.

units of electricity. Trade in electricity is also affected by capacity constraints; every trader has a finite maximum capacity of electricity that they can generate or purchase for resale.

In these experiments, the number of sellers, NS, is the same as the number of buyers, NB. All traders have a capacity of 10 units. All traders are equipped with the (NPT) strategy as described in Section 4.2.4.

10.4.1 Parameters

The NPT strategy is configured with parameters:

$$\begin{aligned} \forall_i \, RE_{s_i} &= 1 \\ \forall_i \, RE_{\rho_i} &= 0.1 \\ \forall_i \, RE_{\mu_i} &= 0.2 \end{aligned}$$

Our design objective is to increase the efficiency of the market, whilst simultaneously keeping the market-power, the degree to which they can control the trade price, of both buyers and sellers to a minimum—we want to increase global profit but without giving unfair advantage to either buyers or sellers. To do this we need to measure efficiency and market power and I have adopted the three variables used in NPT, namely: *market efficiency*, *seller market-power* and *buyer market-power*, as defined in Section 6.3. To recap, market efficiency, EA, is defined as:

$$EA = 100 \left(\frac{PBA + PSA}{PBE + PSE} \right) \tag{10.1}$$

PBA and PSA are the profits that the buyers and sellers, respectively, actually make. PBE and PBE are the profits theoretically available to buyers and sellers, respectively, in an market where all traders bid truthfully and an optimal allocation is made. (We can, of course, compute the result of agents bidding truthfully since we have access to their private values outside the simulation.)

Buyer market-power, MPB, is defined as the difference between the actual profits of buyers, PBA, and the potential equilibrium profits PBE for buyers, expressed as a ratio of the equilibrium profits.

$$MPB = \frac{PBA - PBE}{PBE} \tag{10.2}$$

Seller market-power is computed in the same way:

$$MPS = \frac{PSA - PSE}{PSE} \tag{10.3}$$

Market efficiency, EA, tracks how good our mechanism is at generating *global profit*, whereas the market-power indices, MPB and MPS track to what extent each group is better or worse off compared to the ideal market.

Strategic buyer market power $SMPB$ measures the difference between the actual profits of the buyers and the profits they would get if they bid truthfully in the current market (as opposed to the ideal market assumed when calculating equilibrium profits), expressed as a fraction of equilibrium profits:

$$SMPB = \frac{PBA - PBT}{PBE} \tag{10.4}$$

Strategic seller market-power is computed in the same way:

$$SMPS = \frac{PSA - PST}{PSE} \tag{10.5}$$

Zero strategic market-power values strongly suggest that the mechanism is strategy proof— i.e., there is no way for a given trader to systematically generate profits at the expense of the other traders.

We normalise each variable by mapping it onto the range $[0, 1]$, where 1 represents the optimal value of a variable and 0 represents the worst value. Variables are mapped using the following functions:

$$\widehat{EA} = \frac{EA}{100} \tag{10.6}$$

$$\widehat{MPB} = \frac{1}{1 + MPB} \tag{10.7}$$

$$\widehat{MPS} = \frac{1}{1 + MPS} \tag{10.8}$$

$$\widehat{SMPB} = \frac{1}{1 + SMPB} \tag{10.9}$$

$$\widehat{SMPS} = \frac{1}{1 + SMPS} \tag{10.10}$$

Given these, our aim is to perform a multi-objective optimisation of efficiency and market power. For these initial experiments I combine our different objectives in a simple linear sum with fixed weightings and optimise the scalar fitness value for the particular case where we give equal weighting to efficiency and market-power[5]. Since we have two measures of market power we have two values to optimise:

$$F = \frac{\widehat{EA}}{2} + \frac{\widehat{MPB} + \widehat{MPS}}{4}$$

$$V = \frac{\widehat{EA}}{2} + \frac{\widehat{SMPB} + \widehat{SMPS}}{4}$$

For now, we restrict our search of the mechanism design space to the *transaction pricing rule*, which sets the price of any given transaction as a function of the *bid* and *ask* prices submitted by buyers and sellers respectively. NPT uses a discriminatory-price

[5]The ultimate goal, however, for future work is to use multi-objective evolutionary algorithms to explore the full Pareto frontier.

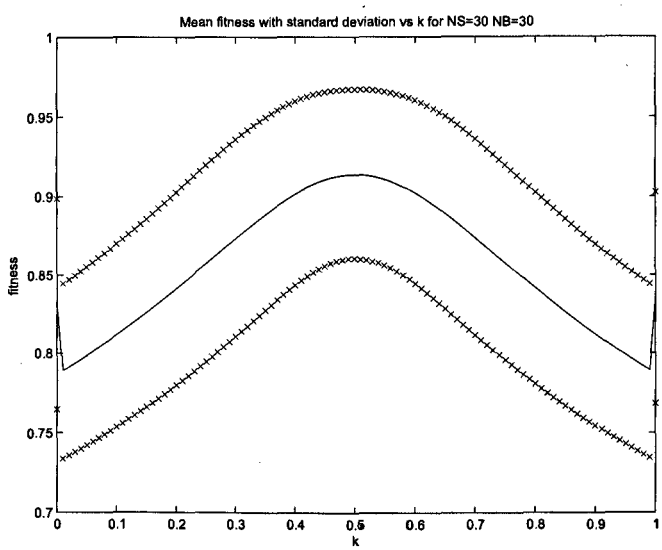

Figure 10.1: Fitness F (with standard deviation) plotted against $k$for a market with 60 traders.

k-double-auction transaction pricing rule (Section 3.3.2), in which a different transaction price is awarded for each matched bid-ask pair in the current auction round. Recall that transaction prices are governed by a parameter k. In the original NPT experiments k is taken to be 0.5.

Our aim is to investigate if there are alternatives to the $k = 0.5$ discriminatory pricing rule that perform well, not necessarily under equilibrium conditions, but when agents play Roth-Erev derived strategies; i.e., adaptive strategies derived from a *cognitive* model of human game playing.

In these experiments, I shall consider the space of all possible pricing rules that are functions of the individual ask price p_a and bid price p_b. Each function is represented as a Lisp s-expression, and Koza's genetic programming [79] is used to search this space. Individual mechanisms are compared according to the criteria represented by F in order to judge their fitness, thus we are using genetic programming to solve a multi-objective optimisation problem. I return to the full details of the GP experiment in Section 10.5.2.

One might ask why we are using genetic programming to search such a vast space, when we could simply restrict attention to the k-double-auction pricing rule, and search for optimal values of k. The reason we use genetic programming is that I see this as a general method of representing *arbitrary* mechanism rules, not just those that can be neatly parameterised. In this particular case, we have chosen an aspect of the auction design that can be simply parameterised, so that I can compare the performance of the genetic programming search against a brute-force search of different values of k. In the following section I use a brute-force search of k to get an approximate view of the

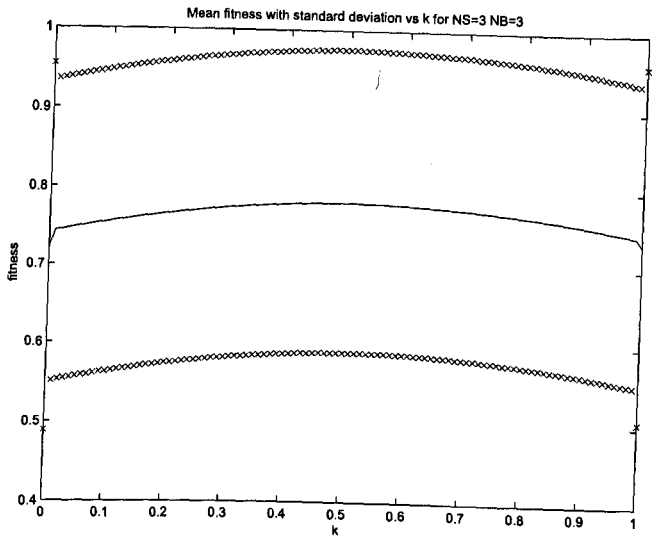

Figure 10.2: Fitness F (with standard deviation) plotted against k for a market with 6 traders.

fitness landscape that our genetic programming search will encounter.

10.5 Experimental results

In this Chapter I report on two aspects of the experimental work I have been carrying out within the electricity market scenario. First I describe work to map out the fitness landscape in which the pricing rule is evolving. We do this by assuming a k-double auction and then calculating the efficiency of the market for different values of k. Secondly, I describe an experiment in which the pricing rule was free to evolve and show that it converged on the k-double auction rule with $k = 0.5$.

10.5.1 Mapping the landscape

Two mappings of the fitness landscape were carried out with 100 different values of k at increments of 0.01. In the first mapping, each auction was run for 100 rounds, and for each value of k we ran 1000 auctions each with a different supply and demand schedule. These schedules were constructed by assigning each agent a random private value from a uniform distribution in the range [30, 1000]. The market variables under observation are averaged over these 1000 different schedules. Figure 10.1 shows the mean fitness measure F for each value of k when the market consists of 60 traders (30 buyers and 30 sellers) and Figure 10.2 shows the mean fitness measure F for each value of k when the market consists of 6 traders (3 buyers and 3 sellers)

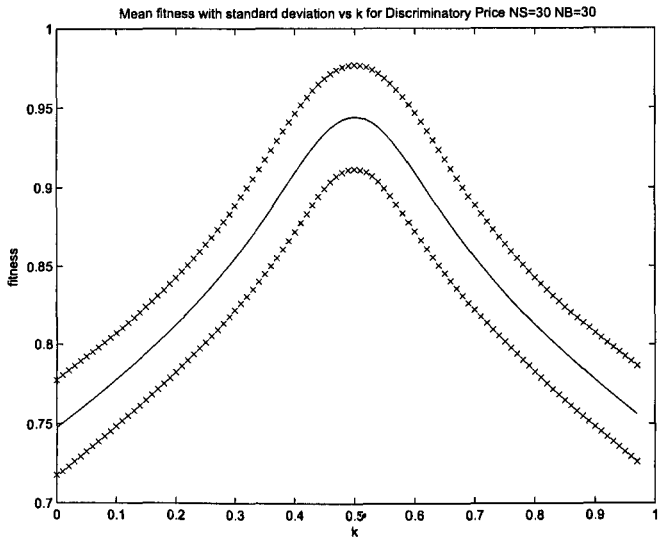

Figure 10.3: Fitness V (with standard deviation) plotted against k for a market with 60 traders.

In the second mapping I looked at fitness measure V. This time, each auction was run for 1000 rounds and outcomes were averaged over 10^5 supply and demand schedules. The results of this mapping is given in Figures 10.3 and 10.4 for 60 traders (30 buyers and 30 sellers) and 12 traders (6 buyers and 6 sellers) respectively. For the second mapping we also looked at the measures of strategic buyer and seller market power. These are shown in Figures 10.5 and 10.6 and suggest that overall strategic market power (the sum of the buyer and seller figures) is approximately zero for $k = 0.5$.

These mappings at different values of k give us an idea of the fitness landscape for the electricity scenario when using our measures of fitness. A qualitative interpretation of this data would suggest that values of k close to 0.5 should be selected by any technique that is applying the k-double auction rule and attempting to learn the best value of k while using our fitness measures. These results suggest that the CH is "heuristically incentive-compatible" for *both* buyers *and* sellers for values of k close to $k = 0.5$.

10.5.2 Evolving pricing rules

Having established the fitness landscape assuming the k-double auction rule, I then set out to search the entire space of possible pricing rules using genetic programming. Each rule was represented as a Lisp s-expression, and I used Koza's basic genetic programming [79] with the parameters given in Table 10.1 to search this space. I made

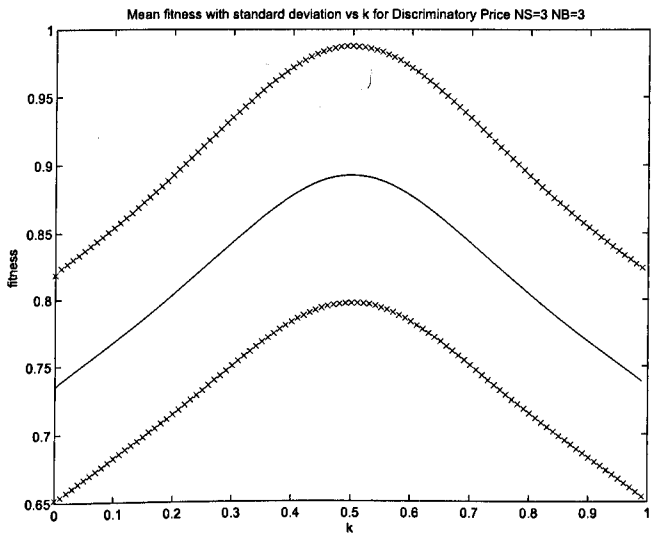

Figure 10.4: Fitness V (with standard deviation) plotted against k for a market with 12 traders.

use of a Java-based evolutionary computation system called ECJ.[6] ECJ implements a strongly-typed GP [92] version of Koza's [79] original system. For the GP experiments in this chapter, the standard Koza parameters were used in combination with the standard Koza GP operators, with the addition of a small amount of parsimony pressure (applied with probability 0.005) in order to counter the effects of GP code bloat.

The function-set consisted of the terminals $ASKPRICE$ and $BIDPRICE$, representing ask price and bid price respectively, together with the standard arithmetic functions, + - * /, and a terminal representing a double-precision floating point ephemeral random constant in the range $[0, 1]$. Thus all we assumed about the pricing function is that it was an arithmetic function of the bid and ask.

Individual mechanisms were compared according to the criteria represented by F in order to judge their fitness during the evolutionary process. As in Section 10.5.1, market outcomes for each pricing rule were computed by simulating agents equipped with the Roth-Erev learning algorithm. I used the same numbers of buyers, 30, and sellers, 30, and 100 auction rounds, but with only 100 different supply and demand schedules, constructed by assigning agents different private values, drawn randomly from a uniform distribution in the range $[30, 1000]$, to evaluate each generation of each population of pricing rules. I ran fewer rounds than in the landscape experiment because, as is usual for evolutionary methods, we had to use many generations and large populations—running each of these for 10^4 supply and demand schedules would have taken a prohibitive amount of time.

Figure 10.7 shows part of the actual pricing rule that was evolved after 90 gener-

[6] http://www.cs.umd.edu/projects/plus/ec/ecj/

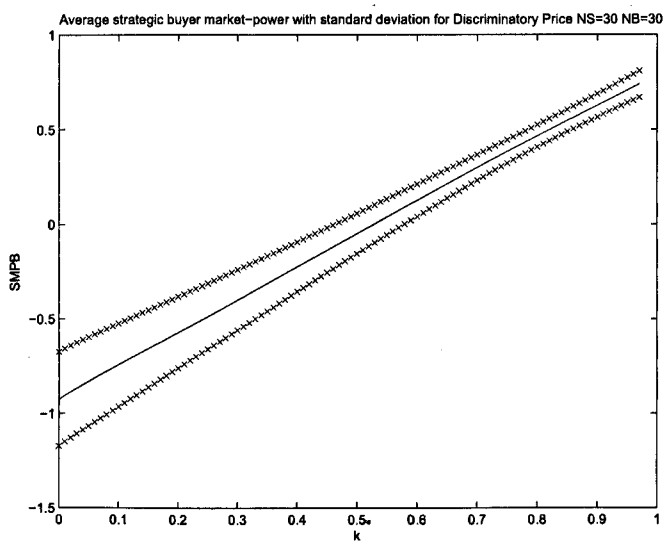

Figure 10.5: Strategic buyer market power plotted against k for a market with 60 traders.

ations. This has been algebraically simplified, but as can be seen it is still far from straightforward, something that is not surprising given the way that standard genetic programming approaches handle the evolution of a program. Plotting the surface of the transaction price as a function of p_b and p_a, given in Figure 10.8, and comparing it with the surface for:

$$0.5p_a + 0.5p_b$$

(given in Figure 10.9) shows that these two functions are approximately equal apart from a slight variation when the ask price is very small or when the ask price is equal to the bid price. Thus the experiment effectively evolved a pricing rule for a discriminatory-price k double auction with $k = 0.5$ from the space of all arithmetic functions of ask and bid price.

Although the fitness landscape for this benchmark problem is very simple, this is a means of validating our design technique before we move on to more complex scenarios.

10.6 Discussion

These results suggest that the approach I am adopting is a reasonable one— I have managed to evolve a rule which not only provides a high fitness, but also generates a rule that, in terms of the prices it sets, is close to a well established rule from the economics literature. The results also support the existing k-double auction rule since our GP search through the space of all functions of the bid and ask price has converged

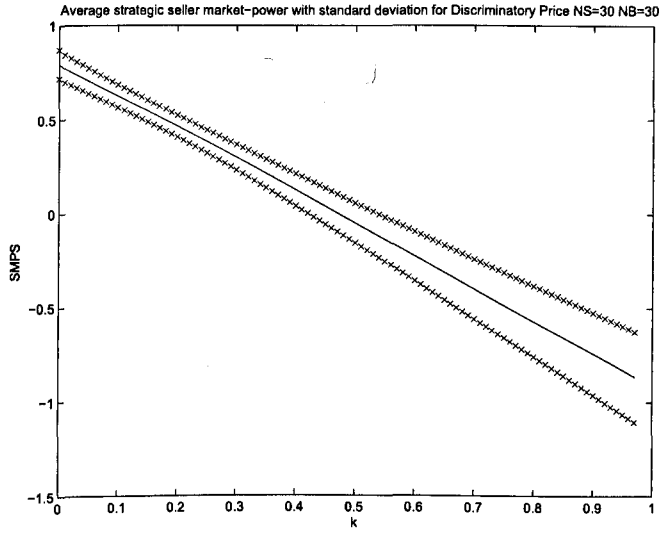

Figure 10.6: Strategic seller market power plotted against k for a market with 60 traders.

$$((0.6250385(0.93977016(ASKPRICE + 0.76238054)))+ \\ (((((-0.19079465)/(ASKPRICE - (((BIDPRICE \\ +BIDPRICE)/(((((ASKPRICE - 1) + 1.6088724)/ \\ (((1 - ASKPRICE) - (ASKPRICE/ASKPRICE))+ \\ (2.5486426 + (BIDPRICE+ \\ 0.000012302072)))) + ((BIDPRICE/ASKPRICE) \\ +((BIDPRICE + BIDPRICE) \\ +(1.430315)/(BIDPRICE \cdot ASKPRICE)))))ASKPRICE))\ldots$$

Figure 10.7: The first few terms of the derived pricing rule.

on a version of the k-double auction rule. This is in contrast to the results obtained by Cliff [26, 27], which discovered a new form of auction between classical buy-side and sell-side auctions.

Interestingly, this result also sheds some light on a problem that was encountered with the approach in [108] when I used genetic programming for both evolving auction rules and evolving trading strategies. In those experiments we noticed that k-double auction pricing rules were evolved early on, when the strategies used by the traders were poor, but did not thrive. It seems it is possible that k-double auction rules do well provided that they are used in auctions with fairly good traders—in auctions with poor

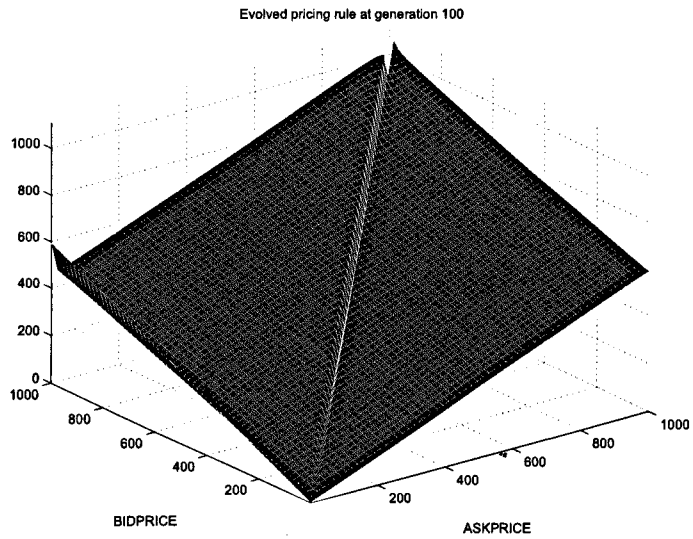

Figure 10.8: The transaction price set by the evolved auction rule.

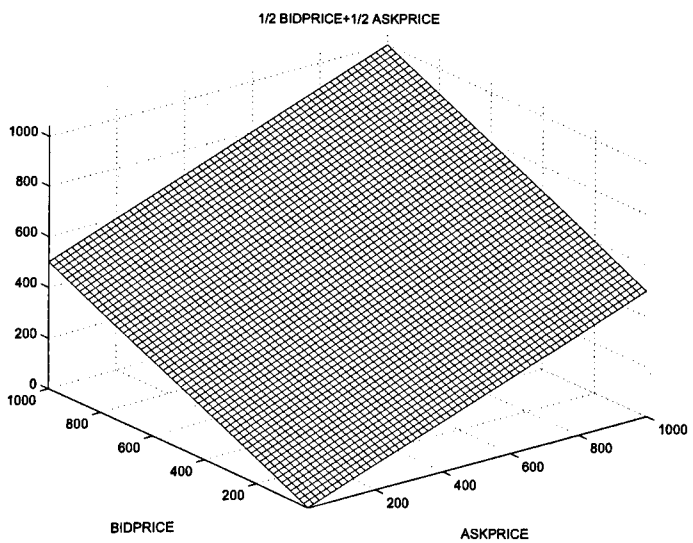

Figure 10.9: The transaction price set by the rule $0.5p_a + 0.5p_b$.

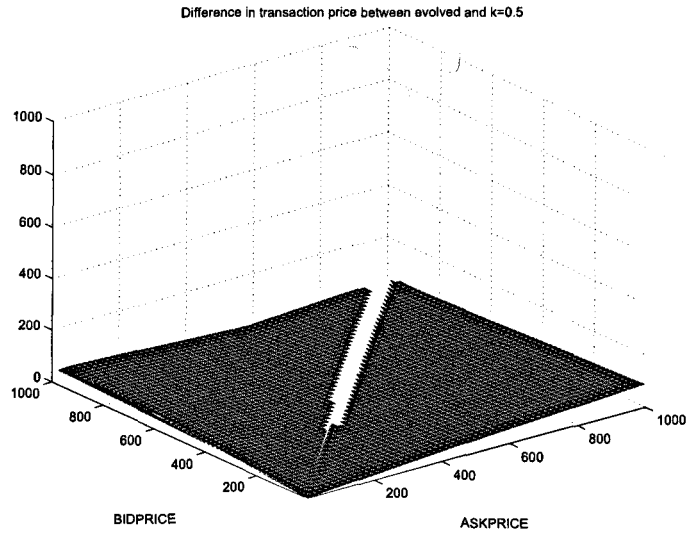

Figure 10.10: The difference in transaction prices between the $k = 0.5$ rule and the evolved rule

traders other rules, which are incompatible with good traders, do better.

This is consistent with a recent view proposed by Philip Mirowski [90, pp. 536–545] of economic marketplaces as complex ecologies. Some markets, such as garage sales, have relatively simple rules and procedures, while others, such as financial futures markets, are, by comparison, very complex. Yet all manage to co-exist, with each type of market, apparently, finding its own niche in which to survive and prosper. Indeed, the oldest markets have survived for hundreds of years without rules from the newer ones being adopted in them. The behaviours of the participants in the different markets are, as one would expect, different. One challenge for computational

Parameter	value
Population size	4000
Selection	Parsimony Binary Tournament
Cross-over probability	0.9
Reproduction probability	0.1
Parsimony size probability	0.005
Cross-over maximum tree depth	17
Grow maximum tree depth	5
Grow minimum tree depth	5

Table 10.1: Koza GP parameters

economics, says Mirowski, is to explain this diversity, how it has arisen and how it is maintained.

10.7 Summary and Contribution

Evolutionary mechanism design, as introduced in this thesis, is an iterative methodology for refining the design of a market mechanism in response to repeated observations of the real life market (*in vivo* analysis) and analysis based on game-theory and simulation (*in vitro* analysis). The methodology is outlined by the following pseudo-code:

input : A set of initial heuristic strategies S, and a legacy mechanism μ
1 **repeat**
2 $S \leftarrow$ `FiSH+` (S, μ) ;
3 *publicise S to participants*;
4 $\vec{x} \leftarrow$ *frequency of each strategy observed in vivo*;
5 $S \leftarrow S \cup \{$ *strategies observed in vivo* $\}$;
6 $\Lambda \leftarrow$ *space of feasible variants of* μ;
7 $\mu \leftarrow \arg\max_{\mu* \in \Lambda}$ EvaluateDesignObjectives$(\mu*, S, \vec{x})$;
8 *implement rules defined by* μ;
9 **until forever** ;

In this chapter I have empirically validated that it is possible to compute step 7, and I have searched a subset of the space of mechanism rules Λ that determine the final clearing price as a function of individual bid and ask prices. In so doing, I have demonstrated that a design in which we set the transaction price halfway between bid and ask prices (a $k = 0.5$ discriminatory pricing policy) has desirable properties, despite the fact that this is not an optimal mechanism according to the usual desiderata and assumptions of auction theory.

Chapter 11

Conclusion and Future Work

In this thesis I have introduced an iterative methodology for the design of market mechanisms called *evolutionary mechanism design*. This differs from traditional mechanism design, which is a static analysis based on rigidly defined design objectives, in which a theoretically pristine mechanism is launched into the world and then remains forever in Nash equilibrium stasis.

Evolutionary mechanism design, in contrast, attempts to take an *engineering* approach. It is not theoretically beautiful, but it is able to take into account real-life ugliness: arbitrary multiple design objectives, *dynamic* adjustment to equilibrium, and constant feedback from an *in vivo* mechanism. The process is described by the following pseudo-code:

input : A set of initial heuristic strategies S, and a legacy mechanism μ
1 **repeat**
2 | $S \leftarrow$ FiSH+ (S, μ) ;
3 | *publicise S to participants*;
4 | $\vec{x} \leftarrow$ *frequency of each strategy observed in vivo*;
5 | $S \leftarrow S \cup \{$ *strategies observed in vivo* $\}$;
6 | $\Lambda \leftarrow$ *space of feasible variants of* μ;
7 | $\mu \leftarrow \arg\max_{\mu* \in \Lambda}$ EvaluateDesignObjectives$(\mu*, S, \vec{x})$;
8 | *implement rules defined by* μ;
9 **until forever** ;

We start with an initial, or "legacy" mechanism μ that exists *in vivo*, that is, a real-life market. This is a reflection of economic reality, in that many market places initially emerge in an ad-hoc fashion and are not necessarily designed from the top-down according to strict auction-theoretic principles [90]; in Chapter 3, I analysed a space of μ based on commonly-encountered variants of legacy double-auction mechanisms, and we saw that many variants of these mechanisms do not satisfy the usual desiderata of auction theory.

We then analyse the set of heuristic[1] strategies S that are commonly observed to be in use. This set of strategies may or may not yield clear-cut equilibria; therefore we conduct an *in vitro* analysis in which we use a combination of game-theoretic analysis and a simulation framework such as that discussed in Chapter 5. By combining these tools using an empirical game-theoretic analysis (Chapter 7) we can discover if there are hitherto unknown strategies that could yield more stable equilibria. This process is summarised by the `FiSH+` algorithm introduced in Chapter 9. If our initial equilibria are not stable, the `FiSH+` algorithm will give us a new set of strategies that yield equilibria with larger attractors, and hence more stable equilibria. As discussed in section 9.4, this is analogous to the incentive-compatibility criterion from conventional static mechanism design in that we are attempting to find strategies that are a clear-cut choice for our traders, just as `TT` is an obvious strategy in an incentive-compatible mechanism. However, this is of no use to a mechanism designer unless our new strategies are actually adopted *in vivo*; hence in step 3 we publicise the resulting analysis to the market participants.

Just as with engineering methods for other complex real-world domains, such as software engineering, our analysis cannot be relied upon to be completely accurate and future-proof [10, 6]. Therefore we continually update our analysis in response to feedback from the *in vivo* mechanism: in steps 4 and 5 we compare our predictions with actuality, and update our set of heuristic strategies S and their observed frequency in the population $\vec{x}$. In Chapter 8 I demonstrated how we can take into account $\vec{x}$ when evaluating whether we are likely to meet our design objectives.

The resulting status quo may not be optimal for our purposes; for example, we may be able to improve the likelyhood of achieving certain design objectives, such as market efficiency or liquidity (transaction throughput) by making small adjustments in a subset of the space of mechanism rules, for example by adjusting parameters such as k in the market clearing rules (3.3.1 to 3.3.3). In Chapter 10, I demonstrated empirically that step 7 can be automated using genetic programming.

Thus, in this thesis I have outlined an iterative methodology for mechanism design: *evolutionary mechanism design*, which incorporates both *in vivo* and *in vitro* analysis, and I have introduced methods for the latter which I have empirically validated as summarised above.

11.1 Future work

Full *in vivo* analysis

In this thesis I have concentrated on the purely computational aspects of the method: that is, *in vitro* analysis. In so doing, I have glanced over some of the challenges presented by the *in vivo* analysis of real-life market places, which may be considerable. For example, in Chapter 8, we saw how our design objectives were affected when we considered different weightings over the frequency with which sniping strategies were observed in the existing mechanism. In the case of a strategy such as sniping, it is relatively straightforward to determine which traders are adopting this strategy,

[1] As opposed to pure strategies in the strict game-theoretic sense. See Chapter 7.

provided that one has access to sufficient historical market data, since we can simply look at the timing of agents' shouts; Roth and Ockenfels [120] provide just such an analysis of the eBay marketplace, which validates that steps 4 and 5 can be performed *in vivo* in the case of a single class of strategy.

However, inferring the existence of other classes of strategies in a real market presents a significant challenge, not least because the true valuation of each agent is not directly observable. Without any prior knowledge of an agent's valuation, it is very difficult to infer whether they are using a strategy even as simple as TT (section 4.1.1). That is not to say, however, that making inferences about valuations is impossible, especially from the privileged vantage point of the agent controlling the mechanism, who potentially has full access to the history of traders' interactions with the market. We may, for example, be able to infer bounds on an agent's valuation by analysing the order statistics of their trade prices over small time periods; or by analysing their trading behaviour in alternative markets for the same commodity; or, in the case of an ascending auction format such as eBay, by observing the price at which runner-up bidders drop out of the auction. With estimates of valuations in hand, it would possible in many cases to infer an agent's strategy. The reverse-engineering of valuations and strategies from market data is a promising area of research, both for those seeking to make profit, as well as for economists seeking to understand the dynamics of real-world marketplaces, and there is an emerging literature in this area [42, 41] to draw upon.

Although it might be impractical in the context of an academic research programme to apply these *in vivo* methods in the context of a market such as a stock exchange, it may be possible to apply them to a markets such as the University of Iowa prediction markets [136]. Prediction markets are exchanges with unique design considerations [156], and an interesting possible research programme would be to conduct a full *in vivo* case study of the application of evolutionary mechanism design to a real-life prediction market through several iterations of the design cycle.

Competing mechanisms

The focus I have taken in this thesis is evolutionary mechanism design as a sequential refinement of a single market institution. However, in many real-life scenarios, multiple competing mechanisms exist simultaneously and offer exchange services for the same commodity. For example, it is often possible to find the same commodities posted on both the eBay[2] and amazon[3] auction web sites. In this scenario, mechanism designers must take into account what rules their competitors are adopting in order to maximise their own design objectives, and mechanism design becomes a competitive interaction. The research conducted by Roth and Ockenfels [120] (discussed above) specifically discusses how amazon and eBay's choice of auction ending rules affects the other competitor. In this scenario, mechanism design is a form of strategic interaction between institutions, as discussed in section 10.2, and it is possible that earlier work in which I used evolutionary computing to co-evolve mechanisms and trading strategies [108, 107] would be useful for *competitive* mechanism design.

[2] `http://www.ebay.com/`
[3] `http://www.amazon.com/`

This kind of analysis may also be useful as an explanatory tool for economists seeking to understand the variety of market institutions observed in nature, from fixed price retail markets to sophisticated electronic exchanges with a bewildering array of trading rules, some of which appear to be based more on historical precedent than rational design considerations. As discussed in the previous chapter, Mirowski [90] suggests that one of the challenges for computational economists is to explain this diversity; how it has arisen and how it is maintained. One possible approach to such an analysis is to think of mechanisms and strategies as co-evolving entities. Under this analysis, the varieties of market institution that we see today correspond to the resulting *equilibria* of the co-evolution between mechanism and strategy. As discussed in section 10.2, the reason that we observe that the majority of institutions for retail goods are fixed-price (for example, high-street retail outlets) rather than dynamically-priced (for example, eBay), might be the same reason that earlier co-evolutionary experiments arrived at fixed-price solutions (section 10.1); fixed-price mechanisms are a best-response to truly "zero-intelligence" strategies which do not bother to bid, and vice versa. Thus the Nash equilibria of the mechanism versus trader game, of which there may be many, may well correspond to the trading formats that we observe to persist *in vivo*. This is a topic that I have only touched upon in this thesis, but could form the basis of interesting future research.

11.2 Applications to other domains

Multi-agent Systems

The focus of this thesis has been the double-auction domain. As discussed in Chapter 2, the double-auction is an important benchmark problem for mechanism design and strategy acquisition. However, my main motivation in this research was to develop techniques that are applicable to the wider field of multi-agent systems technology. As we saw in Chapter 1 one of the principle problems in this field is the engineering of *open* systems. The internet is one of the most complex open systems in existence, and it is increasingly realised that incentive engineering is key in this domain. For example, Friedman and Shenker [52] describe contention over network bandwidth (congestion) by different self-interested parties (agents) in terms of a strategic game, and propose a non-monetary *mechanism* in which socially desirable outcomes can be achieved even when agents follow self-interested strategies by imposing a "handicap" in the form of a network latency that is proportional to an agent's stated bandwidth requirements (analogous to the "handicap" in form of payment that an auctioneer imposes on stated preferences for goods — see Section 3.1.3). It is this kind of *ad-hoc* scenario to which evolutionary mechanism design may be particularly suited, since, in principle, evolutionary mechanism design methods can be used to craft new mechanisms *on the fly*, and in situations in which classical game-theoretic or auction-theoretic assumptions are violated; in the Friedman and Shenker scenario for example, the negative payoffs ("payments") inflicted by the "auctioneer" may be subject to environmental noise, which is not taken into account by existing mechanism designs. This is an ongoing area of research that is being taken up by several research groups: for example, by

the EPSRC *Market-based control of complex computational systems* project[4], and has applications to many other control problems in computer science and beyond, such as scheduling systems [30], memory allocation, and even air-conditioning [24].

From an economic perspective, market mechanisms are traditionally thought of as tools for achieving socially desirable outcomes between agents whose interests are not *necessarily* aligned, and who therefore attempt to maximise only their own utility. However, this does not prevent them from being useful in scenarios where agents' interests *are* in alignment with other. Many scenarios in MAS involve some form of cooperative problem solving [38], which can involve complex coordination between different subsystems, for example the problem of coordinating movements between different joints or actuators in a robotics scenario [13]. In some scenarios it is possible to design a mechanism for these scenarios that brings about the globally-desired outcome when each agent, or subsystem, solves an entirely *local* decision problem (maximising their utility), thus enabling simpler agent implementations; this area of research is known as market-oriented programming [150], in which evolutionary mechanism design may be able to play an important role.

Multi-agent learning & co-evolution

Finally, it is my tentative hypothesis that one of the key principles in acquiring robust strategies in co-evolutionary scenarios may be in the appropriate design of the *game* underlying agent interactions, rather than focusing solely on the co-evolutionary algorithm itself. For example, we may expect that co-evolutionary interactions in games such as paper-rock-scissors, which admits of a single clear-cut equilibrium solution, to result in lower diversity and robustness of phenotypes than in games such as the double-auction where we have a multitude of potential game-theoretic equilibria. It is possible that some of the methods proposed in this thesis may be useful for automatically assessing and constructing environments (games) which encourage diverse and robust solutions in co-evolutionary interaction; for example, by quantitatively estimating the number of equilibria and their respective basin sizes.

[4] `http://www.marketbasedcontrol.com/`

Appendix A

UML Diagrams

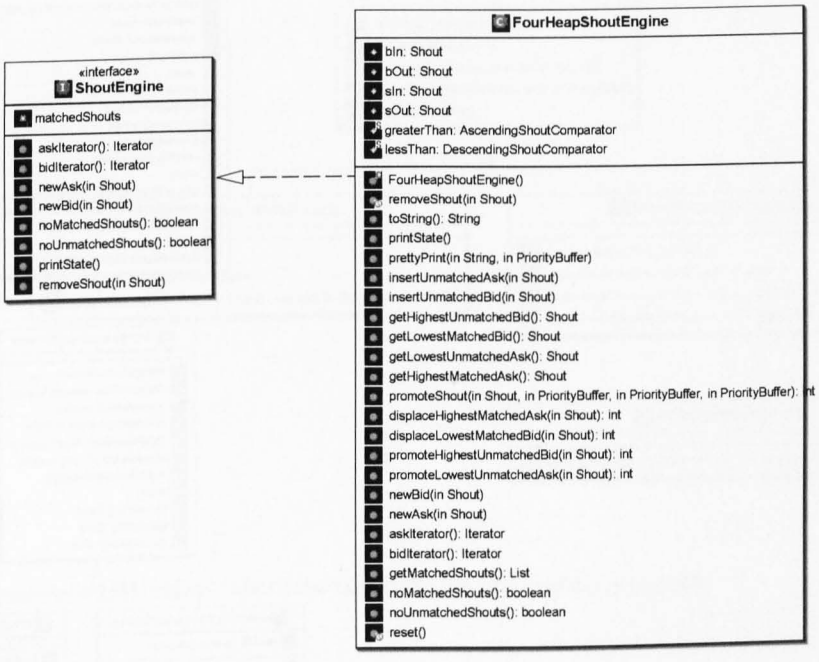

Figure A.1: UML class diagram for the FourHeapShoutEngine class

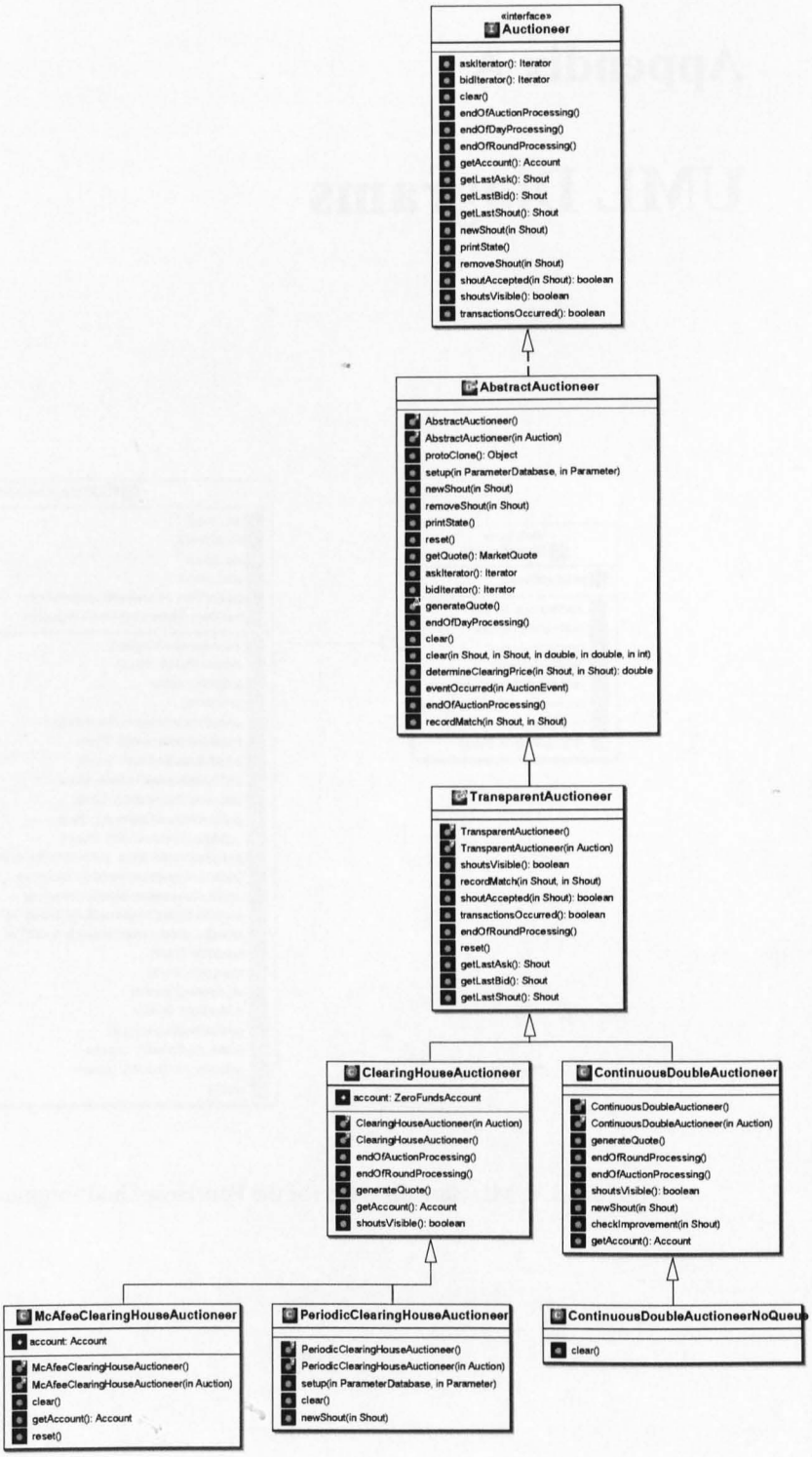

Figure A.2: UML class inheritance diagram for Auctioneer classes

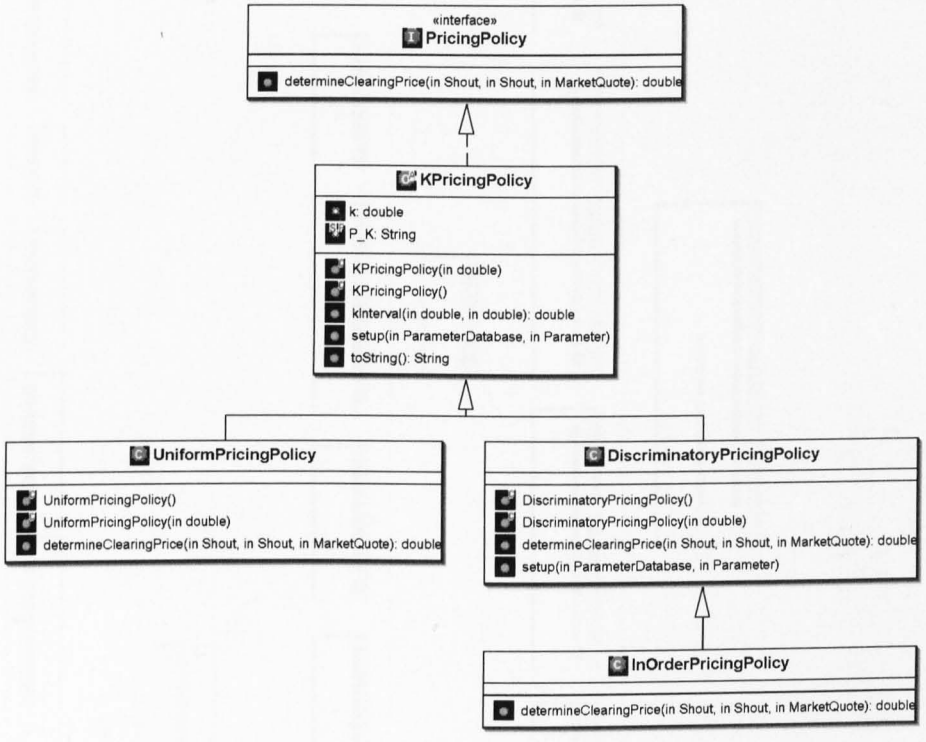

Figure A.3: UML class inheritance diagram for PricingPolicy classes

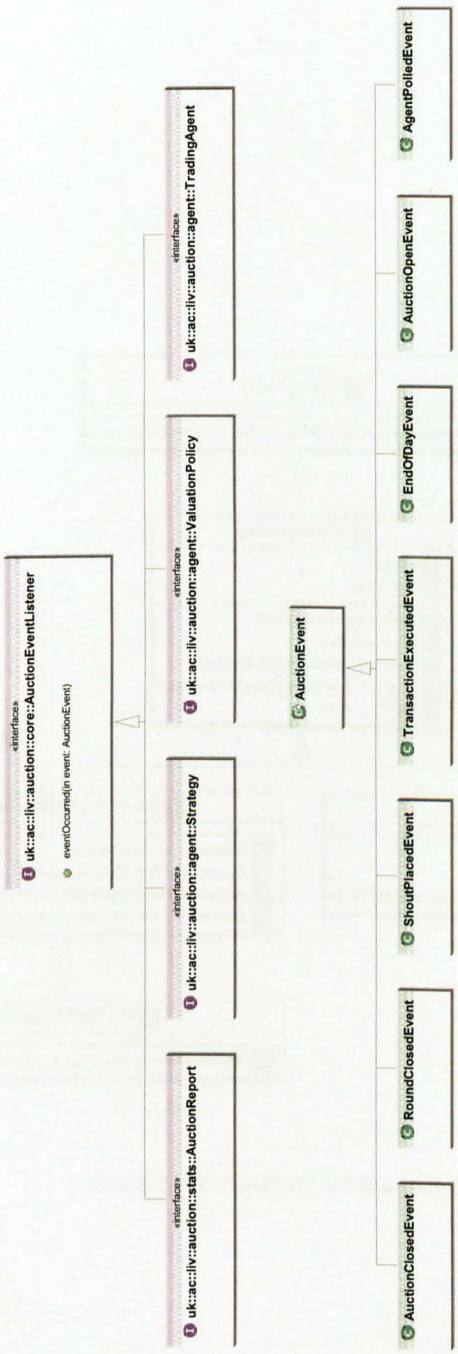

Figure A.4: UML class inheritance diagram for auction events

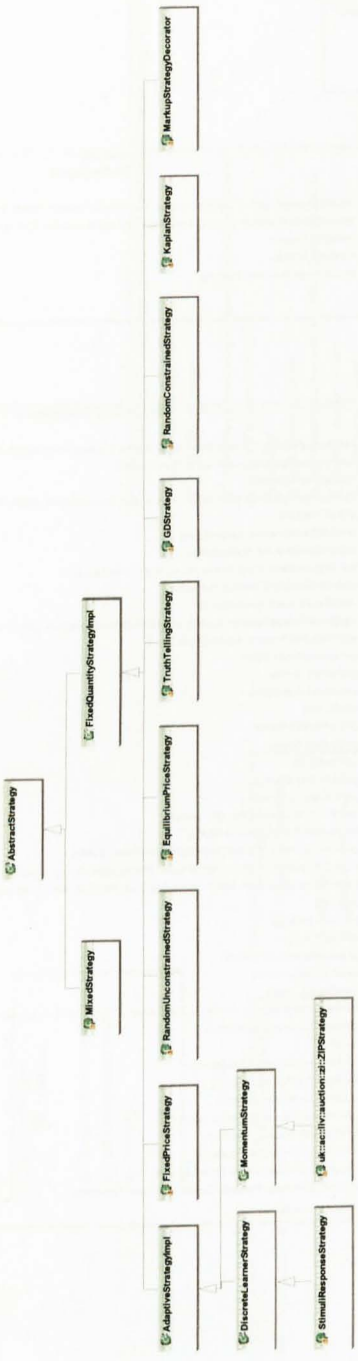

Figure A.5: UML class inheritance diagram for trading strategies

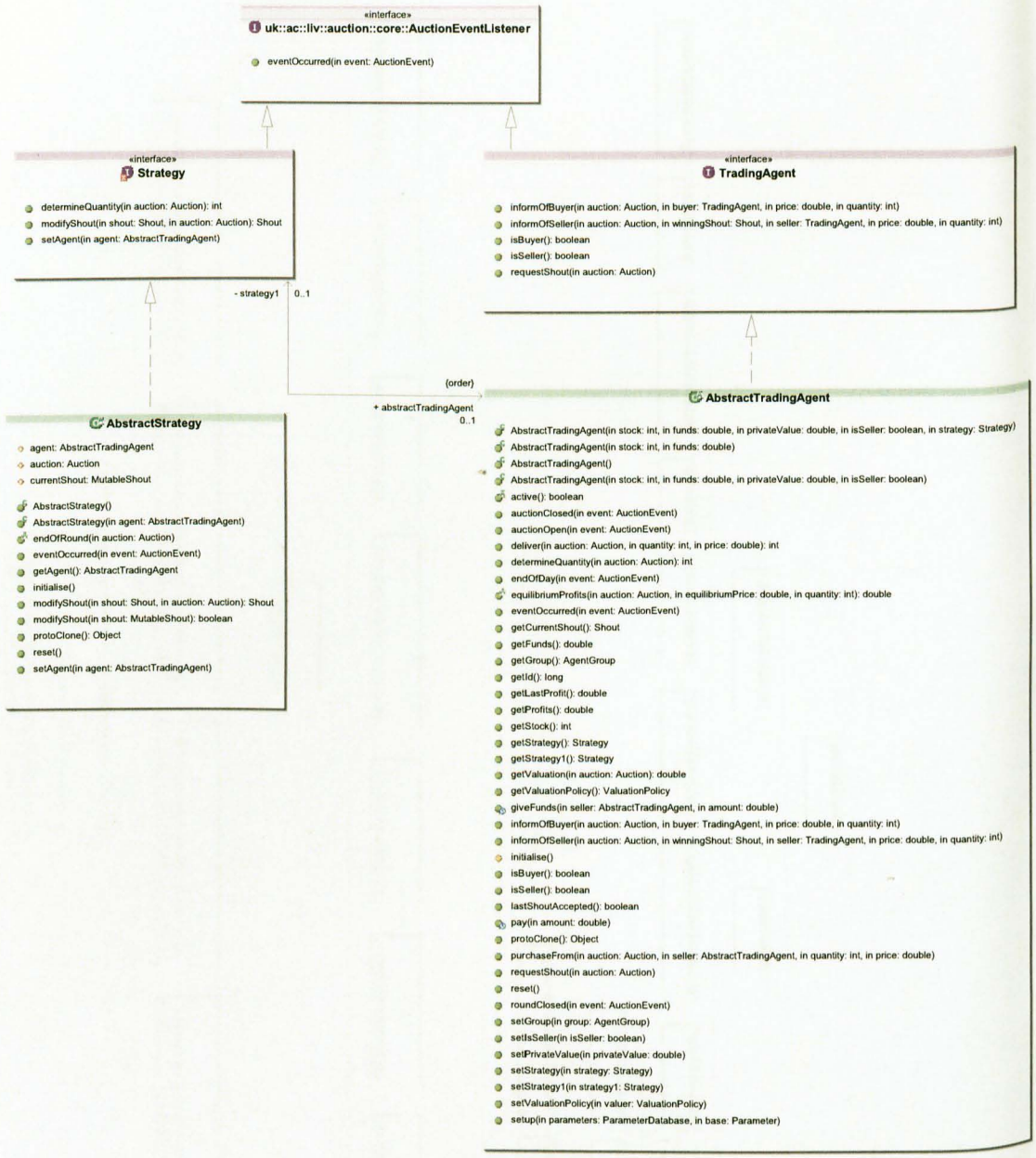

Figure A.6: UML class diagram illustrating relationship between TradingAgent and Strategy

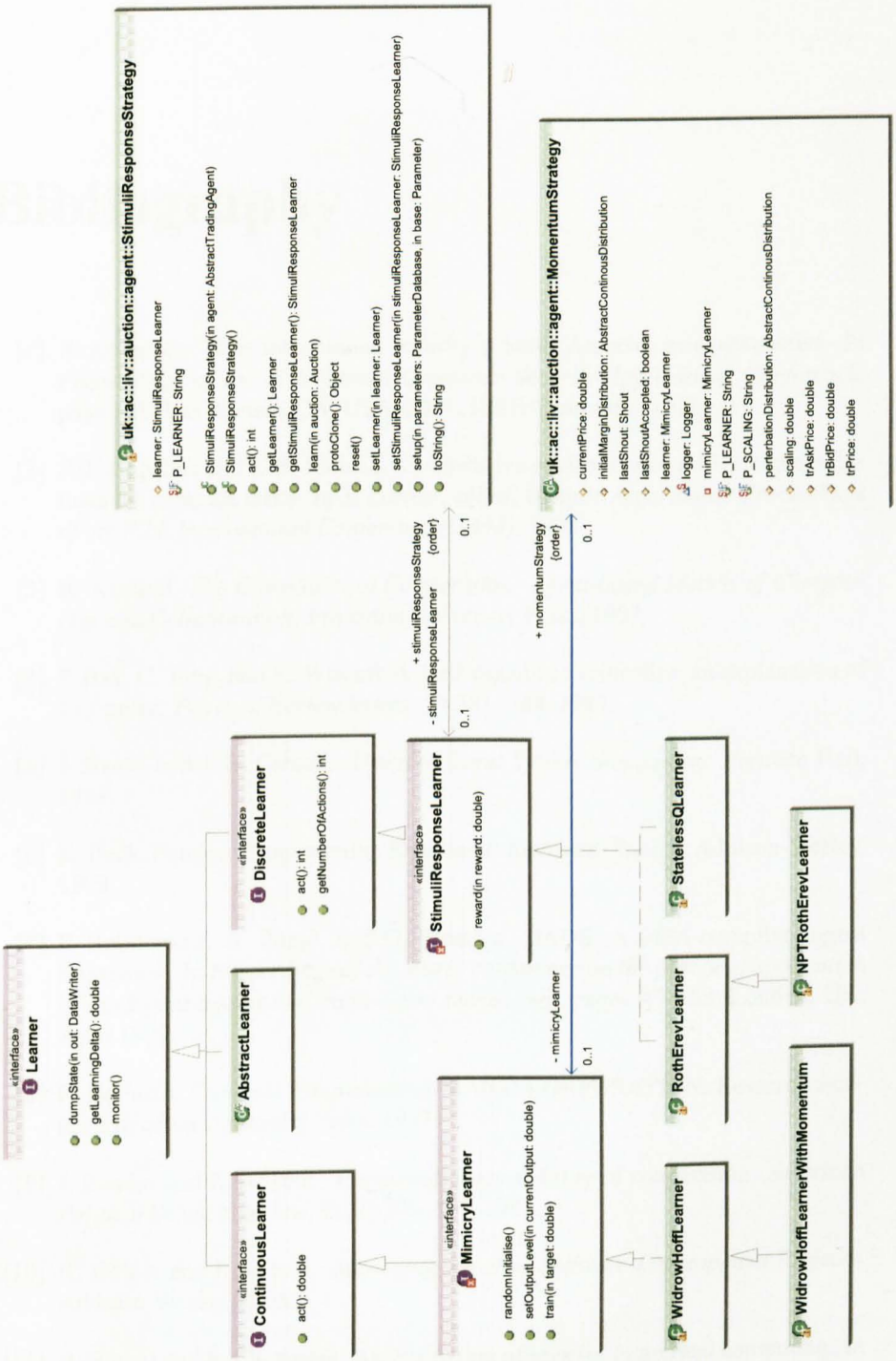

Figure A.7: UML class diagram illustrating relationship between learning algorithms and trading strategies

Bibliography

[1] R. Anderson. Why information security is hard: An economic perspective. In *Proceedings of the 17th Annual Computer Security Applications Conference*, page 358, Los Alimos, CA, USA, 2001. IEEE Computer Society.

[2] P. J. Angeline and J. B. Pollack. Competitive environments evolve better solutions for complex tasks. In S. Forrest, editor, *Genetic Algorithms: Proceedings of the Fifth International Conference (GA93)*, 1993.

[3] R. Axelrod. *The Complexity of Cooperation: Agent-based Models of Competition and Collaboration*. Princeton University Press, 1997.

[4] P. Bak, C. Tang, and K. Wisenfeld. Self-organized criticality: an explanation of $1/f$ noise. *Physical Review letters*, 59:381–384, 1987.

[5] J. Banks and J. S. Carson. *Discrete-Event System Simulation*. Prentice Hall, 1984.

[6] K. Beck. *Extreme Programming Explained: Embrace Change*. Addison-Wesley, 1999.

[7] F. Bellifemine, A. Poggi, and G. Rimassa. JADE: A FIPA-compliant agent framework. In *Proceedings of the fourth conference on the practical application of intelligent agents and multi-agent technology*, pages 97–108, London, UK, April 1999.

[8] R. Bellman. *Dynamic Programming*. RAND CORPORATION. Research studies. Princeton University Press, 1957.

[9] J. Bendor and P. Swistak. The evolutionary stability of cooperation. *American Political Science Review*, 91(2):290–307, 1997.

[10] K. Bittner and I. Spence. *Managing Iterative Software Development Projects*. Addison-Wesley, 2006.

[11] B. Blount and S. Chatterjee. An evaluation of java for numerical computing. In *ISCOPE*, pages 35–46, 1998.

[12] M. E. Blume, J. J. Siegel, and D. Rottenberg. *Revolution on Wall Street: The Rise and Decline of the New York Stock Exchange.* W. W. Norton, first edition, 1993.

[13] C. Boutilier and R. Brafman. Partial-order planning with concurrent interacting actions. *Journal of Artificial Intelligence Research*, 14:105–136, 2001.

[14] C. Boutilier, Y. Shoham, and M. P. Wellman. Editorial: Economic principles of multiagent systems. *Artificial Intelligence*, 94:1–6, 1997.

[15] F. P. Brooks. *The Mythical Man-Month.* Addison-Wesley, 20th anniversary edition, 1995.

[16] R. Brunner and F. Freitag. Elaborating a decentralized market information system. In *On the Move to Meaningful Internet Systems 2007: OTM 2007 Workshops*. Springer Berlin, 2007.

[17] W. D. A. Bryant. Information, adjustment and the stability of equilibrium. Technical Report 6/2000, Department of Economics, Macquarie University, Sydney, Australia, 2000.

[18] 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.

[19] A. Byde. Applying evolutionary game theory to auction mechanism design. In *Proceedings of the ACM Conference on Electronic Commerce 2003*, pages 192–193, 2003.

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

[21] K. Chatterjee and W. Samuelson. Bargaining under incomplete information. *Operations Research*, 31(5):835–851, 1983.

[22] T. W. Christopher and G. K. Thiruvathukal. *High-Performance Java Platform Computing: Multithreaded and Networked Programming.* Prentice Hall, 2000.

[23] G. P. E. Clarkson and H. A. Simon. Simulation of individual and group behavior. *American Economic Review*, 50:920–932, 1960.

[24] S. H. Clearwater, editor. *Market-Based Control: A Paradigm for Distributed Resource Allocation.* World Scientific Publishing Company, 1996.

[25] D. Cliff. Evolving parameter sets for adaptive trading agents in continuous double-auction markets. In *Agents-98 Workshop on Artificial Societies and Computational Markets*, pages 38–47. Minneapolis, MN., 1998.

[26] D. Cliff. Evolution of market mechanism through a continuous space of auction-types. Technical Report HPL-2001-326, HP Labs, 2001.

[27] D. Cliff. Evolutionary optimization of parameter sets for adaptive software-agent traders in continuous double auction markets. Technical Report HPL-2001-99, HP Labs, Bristol, 2001.

[28] 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.

[29] V. Conitzer and T. Sandholm. Complexity of mechanism design. In *Proceedings of the 18th Annual Conference on Uncertainty in Artificial Intelligence (UAI-02)*, pages 103–110, 2002.

[30] M. P. Wellman D. M. Reeves, J. K. MacKie-Mason and A. Osepayshvili. Exploring bidding strategies for market-based shceduling. *Decision Support Systems*, 2005.

[31] C. Darwin. *The Origin of Species*. Gramercy Books, 1998.

[32] C. d'Aspremont and L. Gérard-Varet. Incentives and incomplete information. *Journal of Public Economics*, 11:25–45, 1979.

[33] E. David, A. Rogers, J. Schiff, S. Karus, and N. R. Jennings. Optimal design of english auctions with discrete bid levels. In *Proceedings of the 6th ACM COnference on Electronic Commerce (EC'05)*, pages 98–107, Vancouver, Canada, 2005.

[34] H. Dawid. The convergence of genetic learning in a double auction market. *Journal of Economic Dynamics and Control*, 23:1545–1567, 1999.

[35] R. Dawkins and J. R. Krebs. Arms races between and within species. *Proceedings of the Royal Society of London*, B 205:489–511, 1979.

[36] S. Dreyfus. Richard bellman on the birth of dynamic programming. *Operations Research*, 50:48–51, 2002.

[37] E. H. Durfee. The distributed artificial intelligence melting pot. *IEEE Transactions on Systems, Man, and Cybernetics (Special Section on DAI)*, 21(6):1301–1306, 1991.

[38] Edmund H. Durfee, Victor R. Lesser, and Daniel D. Corkill. Trends in cooperative distributed problem solving. *IEEE Transactions on Knowledge and Data Engineering*, 1(1):63–83, 1989.

[39] P. K. Dutta. *Strategies and Games: Theory and Practice*. MIT Press, 2001.

[40] N. Eldredge and S. J. Gould. Punctuated equilibria: an alternative to phyletic gradualism. In *Models in paleobiology*. Copper & Co., 1972.

[41] J. Engle-Warnick. Inferring strategies from observed actions: a nonparametric, binary tree classification approach. *Journal of Economic Dynamics and Control*, 27:2151–2170, 2003.

[42] J. Engle-Warnick and B. Ruffle. Inferring buyer strategies and their impact on monopolist pricing. Technical Report 2001-W28, Economics Group, University of Oxford, Nuffield College, 2001.

[43] 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.

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

[45] S. G. Ficici and J. B. Pollack. Challenges in coevolutionary learning: Arms-race dynamics, open-endedness, and mediocre stable states. In *Proceedings of of ALIFE-6*, 1998.

[46] S. G. Ficici and J. B. Pollack. A game-theoretic approach to the simple coevolutionary algorithm. In Hans-Paul Schwefel Marc Schoenauer, Kalyanmoy Deb, Günter Rudolph, Xin Yao, Evelyne Lutton, Juan Julian Merelo, editor, *Parallel Problem Solving from Nature — PPSN VI 6th International Conference*, Paris, France, 16-20 2000. Springer Verlag.

[47] S. G. Ficici and J. B. Pollack. A game-theoretic memory mechanism for coevolution. In E. Cantu-Paz et al., editor, *LNCS 2723*, pages 286–297. Springer-Verlag, 2003.

[48] F. M. Fisher. *Disequilibrium Foundations of Equilibrium Economics*. Cambridge University Press, 1983.

[49] M. M. Flood. Some experimental games. Technical Report RM-789, RAND Corporation, Santa Monica, CA, 1952.

[50] D. Friedman. The double auction market institution: A survey. In D. Friedman and J. Rust, editors, *The Double Auction Market: Institutions, Theories and Evidence*, Santa Fe Studies in the Sciences of Complexity, chapter 1, pages 3–26. Westview, 1993.

[51] D. Friedman and J. Rust, editors. *The Double Auction Market: Institutions, Theories, and Evidence (Proceedings of the workshop on double auction markets held June 1991 in Santa Fe, New Mexico)*. Westview, 1991.

[52] E. Friedman and S. Shenker. Learning and implementation on the internet. Departmental Working Paper 199821, Rutgers University, Department of Economics, New Jersey Hall - 75 Hamilton Street, New Brunswick, NJ 08901-1248, September 1998.

[53] D. Fudenberg, M. M. Mobius, and A. Szeidl. Existence of equilibrium in large double auctions. Working paper, Department of Economics, Harvard University, Littauer Center, Cambridge, MA 02138, April 2003.

[54] J. M. Garrido. *Object-oriented Discrete-event Simulation with Java: A Practical Introduction (Series in Computer Systems)*. Kluwer Academic, 2001.

[55] C. Gershenson and F. Heylighen. When can we call a system self-organizing? In *Advances in Artificial Life, 7th European Conference, Dortmund, Germany*, pages 606–614. Springer LNAI 2801, 2003.

[56] H. Gintis. *Game Theory Evolving: A Problem-Centered Introduction to Modeling Strategic Interaction*. Princeton University Press, 2000.

[57] S. Gjerstad and J. Dickhaut. Price formation in double auctions. *Lecture Notes in Computer Science*, 2033:106, 2001.

[58] D. K. Gode and S. Sunder. Allocative efficiency of markets with zero-intelligence traders: Market as a partial substitute for individual rationality. *Journal of Political Economy*, 101(1):119–137, 1993.

[59] J. K. Goeree and C. A. Holt. Ten little treasures of game theory and ten intuitive contradictions. *American Economic Review*, 91(5):1492–1422, December 2001.

[60] P. Gradwell and J. Padget. A comparison of distributed and centralised agent based bundling systems. In *Proceedings of the ninth international conference on electronic commerce*, pages 25–34, Minneapolis, MN, USA, 2007.

[61] A. R. Greenwald and P. Stone. Autonomous bidding agents in the trading agent competition. *IEEE Internet Computing*, 5(2):52, 2001.

[62] L. Gulyas, T. Kozsik, and J. B. Corliss. The multi-agent modelling language and the model design interface. *Journal of Artificial Societies and Social Simulation*, 2(3), 1999.

[63] J. C. Harsanyi. Games with incomplete information played by Bayesian players: Part i. *Management Science*, pages 159–182, 1967.

[64] F. Heylighen. The science of self-organization and adaptivity `http://pespmc1.vub.ac.be/Papers/EOLSS-Self-Organiz.pdf` last accessed on 11/1/2008.

[65] W. D. Hillis. Co-evolving parasites improve simulated evolution as an optimization procedure. In Langton et al., editor, *Proceedings of ALIFE-2*, pages 313–324. Addison Wesley, 1992.

[66] W. Hsu and V. Soo. Market performance of adaptive trading agents in synchronous double auctions. In *Lectures Notes in Computer Science*, volume 2132, pages 108–121. Springer Berlin, 2001.

[67] P. Huang, A. Scheller-Wolf, and K. Sycara. A strategy-proof multiunit double auction mechanism. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pages 166–167, Bologna, Italy, 2002.

[68] A. Hunt and D. Thomas. *Pragmatic Unit Testing in Java with JUnit.* Pragmatic Bookshelf, September 2003.

[69] M. O. Jackson. Mechanism theory. In *The Encyclopedia of Life Support Systems.* EOLSS Publishers, 2000. URL: `www.eolss.co.uk`.

[70] M. O. Jackson and J. M. Swinkels. Existence of equilibrium in single and double private value auctions. Mimeo, California Institute of Technology, 2001.

[71] M. O. Jackson and J. M. Swinkels. Existence of equilibrium in single and double private value auctions. *Econometrica*, 73(1):93–139, 2005.

[72] H. J. Jensen. *Self-Organized Criticality.* Cambridge Lecture Notes in Physics, 1998.

[73] O. Kadan. Equilibrium in the two player, k-double auction with affiliated private values. Technical Report 12.2004, Nota di Lavoro, 2004.

[74] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[75] J. H. Kagel and A. E. Roth, editors. *The Handbook of Experimental Economics.* Princeton University Press, 1995.

[76] P. Klemperer. How (not) to run auctions: the European 3G telecom auctions. *European Economic Review*, 46:829–845, 2002.

[77] P. Klemperer. Why every economist should learn some auction theory. In *Auctions: Theory and Practice.* Princeton University Press, 2004.

[78] J. Kodjabachian and J. Meyer. Evolution and development of control architectures in animats. *Robotics and Autonomous Systems*, 16, 1995.

[79] J. R. Koza. *Genetic Programming: on the programming of computers by means of natural selection.* MIT Press, 1993.

[80] V. Krishna. *Auction Theory.* Harcourt Publishers Ltd., 2002.

[81] P. Krugman. *The Self-Organizing Economy.* Blackwell, 1996.

[82] A. M. Kwasnica and E. Katok. The effect of timing on bid increments in ascending auctions.

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

[84] D. A. MacKenzie. An equation and its worlds: *Bricolage*, exemplars, disunity and performativity in financial economics. *Social Studies of Science*, 33:831–868, 2003.

[85] 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.

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

[87] R. P. McAfee. A dominant strategy double auction. *Journal of Economic Theory*, 56:434–450, 1992.

[88] A. Medio and G. Gallo. *Chaotic Dynamics: Theory and Applications to Economics*. Cambridge University Press, 1992.

[89] N. Minar, R. Burkhart, C. Langton, and M. Askenazi. The Swarm simulation system: A toolkit for building multi-agent simulations. Technical Report 96-06-042, Santa Fe, 1996.

[90] P. Mirowski. *Machine Dreams: Economics Becomes a Cyborg Science*. Cambridge University Press, December 2001.

[91] B. M. Mitnick. The theory of agency: the policing "paradox" and regulatory behavior. *Public Choice*, 24(1):27–42, 1975.

[92] D. J. Montana. Strongly typed genetic programming. Technical Report #7866, 10 Moulton Street, Cambridge, MA 02138, USA, 7 1993.

[93] J. E. Moreira, S. P. Midkiff, M. Gupta, P. V. Artigas, M. Snir, and R. D. Lawrence. Java programming for high-performance numerical computing. *IBM Systems Journal*, 39(1):21–56, 2000.

[94] R. B. Myerson and M. A.Satterthwaite. Efficient mechanisms for bilateral trading. *Journal of Economic Theory*, 28:265–281, 1983.

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

[96] D. Neuman, J. Stoesser, A. Anandasivam, and N. Borissov. Sorma building an open grid market for grid resource allocation. In *Grid Economics and Business Models*. Springer Berlin, 2007.

[97] A. Newell and H. A. Simon. Computer science as empirical inquiry: Symbols and search. *Communications of the ACM*, 19(3):113–126, 1976.

[98] 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.

[99] J. Niu, K. Cai, S. Parsons, and E. Sklar. Reducing price fluctuation in continuous double auctions through pricing policy and shout improvement. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 1143–1150, Hakodate, Japan, 2006.

[100] J. Noble and R. A. Watson. Pareto coevolution: Using performance against coevolved opponents in a game as dimensions for pareto selection. In *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-2001*, pages 493–50, San Fancisco, California, 2001. Morgan Kauffman.

[101] M. J. North, N.T. Collier, and J. R. Vos. Experiences creating three implementations of the Repast agent modeling toolkit. *ACM Transactions on Modeling and Computer Simulation*, 16:1–25, 2006.

[102] T. Nylund and R. Runds. Implementing a random device driver under solaris 8. Master's thesis, Department of Computer Science, University of Gothenburg, 2002.

[103] M. J. Osborne and A. Rubinstein. *A Course in Game Theory*. MIT Press, 1994.

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

[105] S. Phelps, M. Marcinkiewicz, S. Parsons, and P. McBurney. A novel method for automatic strategy acquisition in n-player non-zero-sum games. In H. Nakashima, M. P. Wellman, G. Weiss, and P. Stone, editors, *Proceedings of the 5th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2006)*, pages 705–712, Hajodate, Japan, May 2006. ACM.

[106] 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.

[107] S. Phelps, S. Parsons, P. McBurney, and E. Sklar. Co-evolution of auction mechanisms and trading strategies: Towards a novel approach to microeconomic design. In *Proceedings of the Bird of a Feather Workshops, Genetic and Evolutionary Computation Conference*, pages 65–72, New York, July 2002. AAAI.

[108] S. Phelps, S. Parsons, P. McBurney, and E. Sklar. Co-evolutionary mechanism design: A preliminary report. In J. Padget, O. Shehory, D. Parkes, N. Sadeh, and W. E. Walsh, editors, *Agent-Mediated Electronic Commerce IV: Designing Mechanisms and Systems*, pages 123–143. Springer Verlag, July 2002.

[109] S. Phelps, S. Parsons, E. Sklar, and P. McBurney. Using genetic programming to optimise pricing rules for a double auction market. In *Proceedings of the workshop on Agents for Electronic Commerce*, Pitsburgh, PA, October 2003.

[110] S. Phelps, V. Tamma, M. Wooldridge, and I. Dickinson. Toward open negotiation. *IEEE Internet Computing*, 8:70–76, 2004.

[111] J. B. Pollack and A. D. Blair. Co-evolution in the successful learning of backgammon strategy. *Machine Learning*, 32:225–240, 1998.

[112] C. Preist and M. van Tol. Adaptive agents in a persistent shout double auction market. Technical Report HPL-2003-242, HP Laboratories Bristol, 2003.

[113] T. C. Price. Using co-evolutionary programming to simulate strategic behaviour in markets. *The Journal of Evolutionary Economics*, 7:219–254, 1997.

[114] E. S. Raymond. *The Cathedral and the Bazaar.* O'Reilly, 1999.

[115] P. J. Reny and M. Perry. Toward a strategic foundation for rational expectations equilibrium. Working paper, University of Chicago, 2003.

[116] G. E. Ropella, S. F. Railsback, and S.K. Jackson. Software engineering considerations for individual-based models. *Natural Resource Modelling*, 15(1):5–22, Spring 2002.

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

[118] A. E. Roth. The economist as engineer: Game theory, experimentation, and computation as tools for design economics. *Econometrica*, 70(4):1341–1378, July 2002.

[119] 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.

[120] A. E. Roth and A. Ockenfels. Last-minute bidding and the rules for ending second-price auctions: Evidence from ebay and amazon auctions on the internet. *Americon Economic Review*, 92(4):1093–1103, September 2002.

[121] J. Rumbaugh, I. Jaconson, and G. Booch. *The Unified Modeling Language Reference Manual.* Addision-Wesley, 2004.

[122] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach.* Prentice Hall, second edition, 2003.

[123] J. Rust, J. Miller, and R. Palmer. *The Double Auction Market: Institutions, Theories and Evidence*, chapter 6, pages 155–199. Santa Fe Studies in the Sciences of Complexity. Westview, 1993.

[124] A. Rustichini, M. A. Satterthwaite, and S. R. Williams. Convergence to efficiency in a simple market with incomplete information. *Econometrica*, 62(5):1041–1063, September 1994.

[125] D. Samuelson and C. Macal. Agent-based simulation comes of age. *OR/MS Today*, 33(4):34–38, 2006.

[126] T. Sandholm. Automated mechanism design: A new application area for search algorithms. In *Proceedings of the International Conference on Principles and Practice of Constraint Programming (CP)*, 2003.

[127] M. Satterthwaite and S. Williams. The Bayesian theory of the k-double auction. In Friedman and Rust, editors, *The Double Auction Market: Institutions, Theories and Evidence*, Santa Fe Studies in the Sciences of Complexity, chapter 4, pages 199–125. Westview, 1993.

[128] W. Shakespeare. *Romeo and Juliet.* British Library, London, first quarto edition, 1597.

[129] J. Shirazi. *Java Performance Tuning.* O'Reilly, 1005 Gravenstein Highway North, Sebastopol, CA 95472, USA, second edition, January 2003.

[130] B. Skyrms. Chaos in game dynamics. *Journal of Logic, Language, and Information*, 1:111–130, 1992.

[131] V. L. Smith. An experimental study of competitive market behavior. *Journal of Political Economy*, 70:111–337, 1962.

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

[133] A. Somit and S.A. Peterson, editors. *Dynamics of Evolution: The Punctuated Equilibrium Debate in the Natural and Social Sciences.* Cornell University Press, 1992.

[134] M. Spence. Job market signaling. *The Quarterly Journal of Economics*, 87(3):355–374, August 1973.

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

[136] J. Surowiecki. *The wisdom of crowds: why the many are smarter than the few and how collective wisdom shapes business, economies, societies, and nations.* Doubleday, 2004.

[137] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction.* MIT Press, 1998.

[138] G. Tesauro and R. Das. High-performance bidding agents for the continuous double auction. In *Proceedings of the third ACM Conference on Electronic Commerce*, pages 206–209, Tampa, Florida, USA, 2001.

[139] L. Tesfatsion. Agent-based computational economics: growing economies from the bottom up. *Artificial Life*, 8(1):55–82, 2002.

[140] K. Trzec, I. Lovrek, and B. Mikac. Agent behaviour in double auction electronic market for communication resources. In *Knowledge-Based Intelligent Information and Engineering Systems*, pages 318–325. Springer Berlin, 2006.

[141] L. V. Valen. A new evolutionary law. *Evolutionary Theory*, 1:1–30, 1973.

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

[143] W. Vickrey. Counterspeculation, auctions and competitive sealed tenders. *Journal of Finance*, 16:8–37, 1961.

[144] W. Vickrey. Auctions and bidding games. In *Recent Advances in Game Theory*, number 29 in Princeton Conference Series, pages 15–27, Princeton, NJ, 1962. Princeton University Press.

[145] W. E. Walsh, R. Das, G. Tesauro, and J. O. Kephart. Analyzing complex strategic interactions in multi-agent games. In *AAAI-02 Workshop on Game Theoretic and Decision Theoretic Agents*, 2002.

[146] 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*, volume 3048 of *LNAI*, pages 109–123, Melbourne, Australia, 2003. Springer.

[147] J. C. H. Watkins and P. Dayan. Qlearning. *Machine Learning*, 8:279–292, 1992.

[148] R. A. Watson and J. B. Pollack. Coevolutionary dynamics in a minimal substrate. In Lee Spector, Erik D. Goodman, Annie Wu, W. B. Langdon, Hans-Michael Voigt, Mitsuo Gen, Sandip Sen, Marco Dorigo, Shahram Pezeshk, Max H. Garzon, and Edmund Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 702–709, San Francisco, California, USA, 7-11 2001. Morgan Kaufmann.

[149] J. W. Weibull. *Evolutionary Game Theory*. MIT Press, first mit press edition, 1997.

[150] M. P. Wellman. A market-oriented programming environment and its application to distributed multicommodity flow problems. *Journal of Artificial Intelligence Research*, 1:1–23, 1993.

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

[152] M. P. Wellman, D. M. Reeves, K. M. Lockner, and R. Suri. Searching for walverine. In *Proceedings of the IJCAI-05 Workshop on Trading Agent Design and Analysis (TADA-05)*, Edinburgh, Scotland, 2005.

[153] M. P. Wellman, P. R. Wurman, K. O'Malley, R. Bangera, S.-d Lin, D. Reeves, and W. E. Walsh. Designing the market game for a trading agent competition. *IEEE Internet Computing*, 5(2):43–51, 2001.

[154] S. R. Williams. The nature of equilibria in the buyer's bid double auction. Discussion Paper 793, Department of Economics, Northwestern University, 1988.

[155] S. R. Williams. Existence and convergence of equilibria in the buyer's bid double auction. *The Review of Economic Studies*, 58:351–374, 1991.

[156] J. Wolfers and E. Zitzewitz. Prediction markets. *The Journal of Economic Perspectives*, 18(2):107–126, 2004.

[157] M. Wooldridge. *Reasoning about Rational Agents*. MIT Press, first edition, July 2000.

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

[159] 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.

[160] P. R. Wurman, W. E. Walsh, and M. P. Wellman. A parameterisation of the auction design space. *Games and Economic Behaviour*, 35:304–338, 2001.

[161] H. Peyton Young. *Individual Strategy and Social Structure*. Princeton University Press, 2001.

[162] A. Zahavi and A. Zavahi. *The Handicap Principle: A Missing Piece of Darwin's Puzzle*. Oxford University Press, 1997.