ECONOMIC ALGORITHMS FOR THE MANAGEMENT OF RESOURCES IN COMPUTER SYSTEMS

Submitted by Peter Gradwell
for the degree of
Doctor of Philosophy
of the University of Bath
Department of Computer Science
2009

Abstract

Cloud computing and distributed Grid computations in the e-science and commercial spheres are beginning to make accessible huge amounts of computing power with "just in time" availability. However, the economic models surrounding these systems are static and uniform, with charging models that, for web-based cloud systems work on a price per unit per hour basis, whilst for educational type resources, fixed contractual arrangements and multi-year projects are more prevalent.

The common place practice of using just-in-time capacity planning and variable pricing algorithms, such as those pioneered by airlines like EasyJet, tells us that the cost of delivering these services and the price that should be paid for them is a much more complex beast. Future Grid and Cloud Computing computations will be enabled by participants trading resources in order to construct bundles of goods or services in both new commercial arenas and the more well established "e-science" experiments in science, engineering and, now emerging, social sciences.

A combinatorial auction (CA) is a natural choice for determining the optimal allocation for a bundle of required goods and services, but the space and time dimensions that characterise a Grid compute cloud would appear to indicate they are incompatible.

This thesis proposes that an analogue of a physical commodities market is more appropriate for distributed resource allocation and that there is a class of bundling problems whose complexity properties appear to make the utilisation of a CA impractical. We therefore compare the two techniques for resource bundling and investigate the cross-over point, to enrich our understanding of how combinatorial auctions and distributed markets may be used together to improve distributed resource allocation practices.

Acknowledgements

While the work described here is all my own, its completion owes a great deal to many people.

Richard Male at the Southwest RDA was kind enough to recycle my tax dollars into an industrial grant for the first three years of my studies, offsetting my fees and getting me started.

The preliminary implementation of the MDA system was completed as a final year degree project by Gislin Kamda and I am grateful to him for giving our ideas an early breath of life as code.

Much later, I was lucky to work with Michel Oey and Reinier Timmer at VU University, maintainers of the Agentscape project. In addition to kind hospitality, helping re-code my algorithms in the Agentscape framework and working on a joint paper, Michel went beyond the call of duty in analysing my work for correctness. It was only through answering his questions that I felt confident in my work.

In their book "How to get a PhD" Phillips and Pugh suggest that it is fatal for a student to enter employment whilst writing up their thesis. Having already started a small business in December 1998 I decided that warning would not apply—I already had a job!

Since that time the business has grown substantially, multiple millions of pounds have been earned and spent in the local economy and I have directly employed over 40 people, to serve over 14,000 customers. The business has provided a unique set of challenges and diverted me from my studies, both physically and mentally, sometimes welcome, frequently delaying, but importantly it provided the context (and funding) that enabled me to stay the course.

I could not have managed two full time endeavours without the support of my excellent staff, management team and customers who fought the fine line between hiding me from the detail of the business and needing my input to drive them forward. I look forward to being able to focus on you properly.

My family has also done their best to keep me on track and it is a privilege to come from a family that values education so highly. My parents have understood, encouraged and proofread every step of the way. I am glad to beat my middle brother to becoming a medical doctor, whilst the mental bibliography of Philip, my second brother reading Politics, Philosophy and Economics at Oxford has saved me hours of research.

The real reason I wanted to achieve a PhD was to prove to myself that I could solve a difficult, intellectual problem; that I could be as capable as all those whom I admire that already have this qualification. My selfish pursuit has come at great cost to my closest supporters, whom I cannot repay: (i) Julian, my supervisor, for I missed every deadline, (ii) Kate, whom I ignored and could not speak to for my mind was exhausted and (iii) my Grandfather, who did not live to see the conclusions written.

I sincerely thank you all.

Contents

List of Figures

Chapter 1

Introduction to the Problem Space

1.1 Context and Motivation

Alice and Bob are physics researchers looking to submit a large quantity of computations, perhaps processing recent experimental results. Each has different budgets, priorities and resource requirements and given the limited capacity of their local computing cluster a decision about how to allocate the resources must be made. Ideally, the optimal allocation should be chosen.

This problem arises in many resource management and procurement applications. We are specifically interested in Cloud computing and distributed Grid computations because new developments in the e-science and commercial spheres are making huge amounts of computing power accessible with "just in time" availability. However, the economic models surrounding these systems are static and uniform, with charging models that, for web-based cloud systemsi, work on a price per unit per hour basis, whilst for educational type resources, fixed contractual arrangements and multi-year projects are more prevalent.

The common place practise of using just-in-time capacity planning and variable pricing algorithms, such as those pioneered by airlines like EasyJet, tells us that the cost of delivering these services and the price that should be paid for them is a much more complex beast. Future Grid and Cloud Computing computations will be enabled by participants trading resources in order to construct bundles of goods or services in both

new commercial arenas and the more well established "e-science" experiments in science, engineering and, now emerging, social sciences. What mechanism should we use to support the trading and resource allocation?

A combinatorial auction (CA) is a natural choice for determining the optimal allocation for a bundle of required goods and services, because it guarantees to give the best economic outcome (Pareto optimal) for parties concerned. However, whilst the algorithms have been refined through significant research the mode of problem solving is fundamentally NP-Hard and requires both a centralised mechanism and complete revelation of the participants preferences and valuations to work.

An alternative solution would be to distribute the problem. With multiple auctions and trading agents we can remove the inherent single point of failure and concurrently solving a larger number of simpler, linear problems, in a predicable time-frame. However, distributed markets have their own failings and typically do not produce Pareto-optimal allocations which does make them unsuitable for some applications.

In order to understand what makes an outcome optimal and come to an informed decision as to what the appropriate solution is for a given resource allocation problem a greater understanding of their workings, outcomes and the relative performance of each is required.

Through an empirical analysis this thesis compares the two techniques for resource bundling and investigate the cross-over point, to enrich our understanding of how combinatorial auctions and distributed markets may be used together to improve distributed resource allocation practises and to enable the reader to understand the parameters that define which technique is appropriate for a particular class of resource allocation problems.

1.2 Introduction to the Problem Space

We wish to determine which method should be used in order to compute the most appropriate allocation of a generic set of resources for a given set of circumstances and develop an understanding of how the different approaches available relate to one another. Consider three scenarios:

1. We have complete information about the preferences of the mechanism participants; we have a single point of centralised decision making; we wait an infinite amount of time; we produce an optimal outcome. This is known as the combinatorial auction solution.

2. We have incomplete preference information; decentralised decision making, a known (linear to problem size) amount of time to wait and we produce a suboptimal outcome, but one which is useful because it is built out of the preferences expressed by market participants. This would be the problem solved by our distributed market, implemented as the "Multiple Distributed Auctions" (MDA) system.

3. We have the trivial problem. Consider a situation of needing to allocate 30 goods to 10 participants. By rolling a 30 sided dice, we can, through a centralised decision making process allocate the resources to the participants. We have zero preference information, it is a very quick process and we have no information about the quality or usefulness of the outcome.

There are, therefore, a number of axes, or scales:

1. Centralised vs Decentralised Decision Making
2. Complete preference information vs Incomplete preference information
3. Time available / expended on determining the solution
4. Optimal vs Suboptimal allocations (for profit or utilisation)

Not all the axes are sliding, linear scales. Decision making is either centralised or decentralised, with no degrees of variance between, but preferences can be incrementally revealed. Time is a linear scale and we can measure the proximity and quality of solutions through their varying degrees of separation.

These four axes allow us to define our problem space, give relative position to each of our example problems and to highlight the difference between the options.

Consider the question of centralised vs decentralised first: Is a financially driven auction the best choice or should a trading market be used? Grid computing networks are

potentially a major user of distributed resource allocation systems, but the problem is certainly not limited to that domain and there is much literature on the subject outside the Grid computing sphere particularly with respect to distributed algorithms. By way of example, the seminal 1980's paper, "A Micro-economic Approach to Optimal Resource Allocation in Distributed Computer Systems" by Kurose and Simha [KS89] considers the benefits of using an economy-based decentralised algorithm for file system allocation. They argue that price is the ideal valuation function for a distributed market for two reasons:

1. Price can ultimately be seen as a projection function from a multi-dimensional vector of "values" — both quantitative and qualitative — to a single value and;
2. Secondly, in a free market, over the longer term, the price, cost and the marginal value that a user assigns to a good, will all converge as competition drives down price and makes a number of good alternatives available.

Work on similar problems in other related areas (parallel computing, file system scheduling, etc.) [ASGH95, FBK96, SH80] typically seems to focus on a simplified environmental model where a significant weakness is a lack of a free market or "real world" aggressiveness and competition that a financially-based allocation environment (auction, trading floor, etc.) should emulate and we describe in our MDA implementation in Chapter 4.

In our introduction, we considered three approaches to resource allocation. Of these, the dice throwing approach is unsatisfactory as it is neither optimal nor inclusive of preferences (although in some circumstances it is arguably viewed as "fair"). There are two relevant approaches to solving a resource allocation problem, the Combinatorial Auction and a Distributed Market.

1.2.1 The Combinatorial Auction

A combinatorial auction is an auction in which bidders can place bids on combinations or "bundles" of items, rather than just individual goods. They have wide applications for use, in auctions for logistics, radio spectrum and many other procurement scenarios.

Most computational resource allocation problems are resolved through the use of a centralised combinatorial auction. However, this is an NP-Hard compute problem which has scalability limits and which runs in unpredictable time, as illustrated in Figure 1.1. Most solutions to this problem combine tree search with heuristics, so if the heuristics are insufficient to prune the search tree and make the problem space "small enough" the algorithm might not be able to complete the search in the given amount of time.

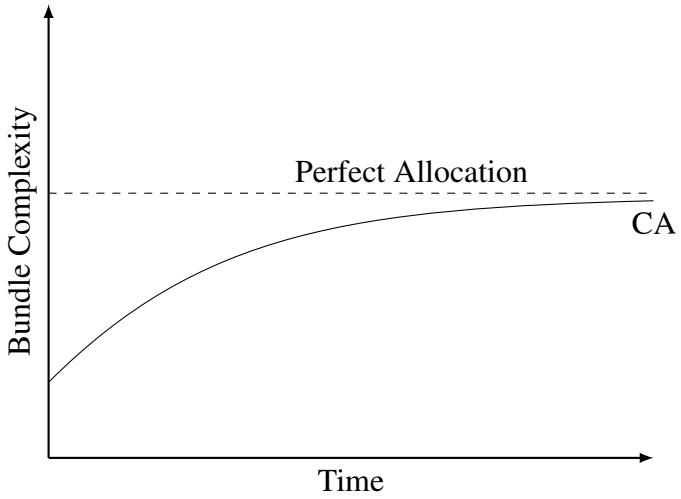

Figure 1.1: Hypothetical Projection of Complexity Model

Looking at a simple set of bundles, we can begin to determine the possible outcomes. For example in Figure 1.2 each line represents a bundle, with the first line showing that agent A will bid 2426 for elements 0, 2 and 3.

Agent	**Budget**	**Requirements**
A	2426	0, 2, 3
B	1969	0, 1, 2
C	32	0
D	781	1, 3
E	370	0, 1, 4

Figure 1.2: Sample CA Bundle Data

A CA solver will build a search tree, similar to Figure 1.3 and use it to determine the answer which will provide the most value for the goods on sale. It does this using a combination of heuristics to prune the tree and various tree search algorithms. In our simple example we can see that selling bundle A, comprising goods 0,2,3 for a value of 2426 achieves the maximum value possible for this auction.

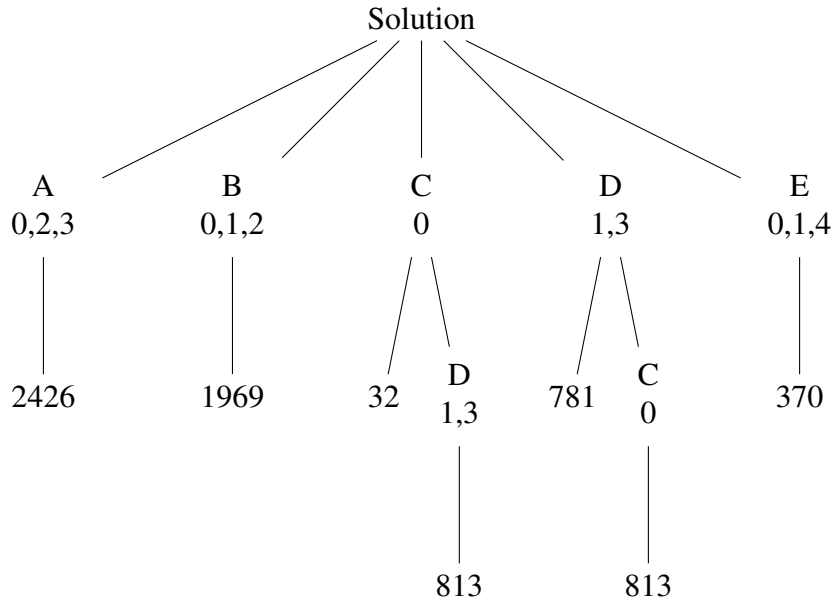

Figure 1.3: Solution Search Tree

1.2.2 Problems with Combinatorial Auctions

When trying to solve the described resource allocation problem, using a centralised market, a number of problems arise:

Robustness: Distributed computer systems inherently have built in redundancy and scalability, as can increasingly be seen in today's cloud computing services. As well as redundancy of the platform, one of the key attractions of scalable computing resources is the ability to quickly increase and reduce usage through the advanced management tools provided. Therefore if we are to provide a robust environment for the execution of applications then we must equally implement management mechanisms which are rapid and robust. Relying on a single, centralised auctioneer to make all the resource management decisions about a platform would not be a sensible or viable solution.

Complete Information: In order to compute the solution to any combinatorial auction, the auctioneer needs to have complete information about all of the preferences for all parties (buyers and sellers) in the market. In a distributed network there are two constraints: (i) Communication between nodes may mean that information for all the potential buyers and sellers is not available at any single point in time. (ii) The structure of a distributed system means that it will encourage participation from buyers and sellers across multiple distributed domains.

This means that for reasons of security and privacy, individual traders may not wish to make their full preferences known to the auctioneer. This is particularly true in a supplier-customer relationship, because the parties will naturally have different motivations and objectives in their negotiations. Note that whilst papers such as those by Sandholm [SG06] do investigate the potential for solving centralised problems with minimal preferences they do not achieve optimal solutions and the proximity of the derived utility to the optimal depends on a number of variables including the number of bids in the auction.

By way of example, the D-CIS Lab at Thales has identified two areas of work which demonstrate applications in which agent based systems are being used to address NP-Hard problems in the real world where a combinatorial approach is infeasible. This is a paradigm they call "Actor-Agent Communities". Firstly, they have provided decision support software to help manage interruptions to train schedules in the Netherlands railway system and secondly to environmental impact decision making in Rotterdam Harbour.

Railway scheduling is a subject that has been long studied and with recent advances in computing power, heuristic development and search algorithms it is common place to find computer systems utilised in the production of initial railway schedules. However, the problem of effectively managing the railway schedule in the event of a disruption to the network is currently unsolved. How could our work on multiple distributed auctions help further the developments of the D-CIS lab in refining their solution to this problem? Firstly, the train rescheduling problem can be characterised as an NP-Hard problem with complex heuristics which requires search to produce a solution. However, when managing the crisis of train schedule interruption it is potentially more important to provide a workable solution quickly than to provide the optimum solution (i.e. one with zero knock on effects to the schedule) because a single knock on effect in the future can be easily managed, where as a current disruption may generate much larger, immediate effects in the network. In an application where the optimal outcome is not necessary, a satisficing [Sim55] solution can be obtained through the use of distributed auctions which may provide a speedy and useful solution.

The question of environmental decision support in Rotterdam Harbour is a similar problem to which we can lend our expertise. The challenge in environmental management is the coordination of response across a very large number of different governmental agencies, all of whom have differing priorities and responsibilities, but whom

must play a coordinated role in responding to an environmental accident.

Representing the available agency resources and desired actions as goods in the Trading Agents model we can package their tasks as bundles and use markets to determine effective strategies for collaborative team working, whilst taking into account a greater number of different constraints and trade offs that are imposed in such a situation, than would be possible if we used a centralised planning solution.

1.2.3 Distributed Market

In order to address these challenges and develop an empirical understanding of the dynamics of decentralised resource allocation systems we built a distributed market, called the Multiple Distributed Auction (MDA)—see Figure 1.4. Initially this was a uni-processor simulation of the MDA using the JASA [PMPM06] and Repast [NCV06] toolkits. The system begins with an 'Oracle' that is boot-strapped from a datafile produced by the "Combinatorial Auction Test Suite" software (CATS) [LBPS00] that provides the goods and bundles needed to run the market. From this the Repast simulator is used to manage a multitude of Agents which participate in a number of continuous double auctions to buy and sell the goods in order to complete their desired bundles.

We do not claim that the concept of distributed markets to be a new or novel idea and the contribution of this thesis is the comparison of centralised and decentralised markets.

Researchers such as Parkes [PS04] have looked extensively at auctions and resource allocation mechanisms with systems such as ICE (Iterative Combinatorial Exchange) [LJC+08], a fully expressive iterative combinatorial auction, and Bellagio [ABC+09] a distributed market scheduler implemented for the Planet Lab grid computing environment. The difficulty that both of these works found however, highlighted in the review of Bellagios effectiveness [ABC+09], was that whilst having a distributed market did increase allocation efficiency there was a real problem of replicating the market's input data because workloads from scheduling environments are, by nature, entirely random and therefore it is difficult to identify the relative performance of the distributed system with the known body of work on centralised systems.

We have additionally argued that it is infeasible to accurately predict the run-time

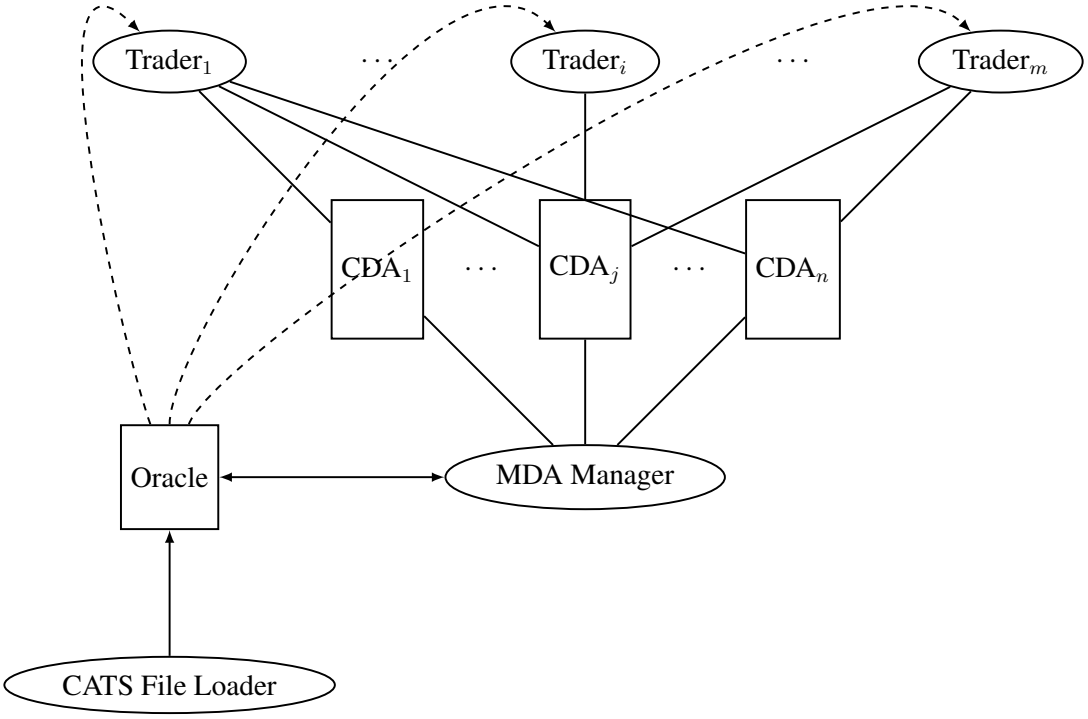

Figure 1.4: Schematic of MDA market structure

behaviour of distributed systems theoretically and in order to make that comparison and hence understand which system is appropriate for the problem the user wishes to solve, it is necessary to have an implementation of both systems, supplied with a consistent set of input data, which is what we have done.

Although authors such as Parkes have published papers on the design and output of their systems, source code is not available and therefore it has been necessary to implement our own system of distributed auctions.

The market comprises an Oracle, many Traders, an MDA manager and many CDAs (Continuous Double Auctions):

Oracle: responsible for handing out bundles to traders on demand. Consequently, varying the Oracle's output rate constrains supply and demand in the market. Bundles can be requested from the Oracle and also later returned to it, if they cannot be purchased/sold. All transactions are reported to the Oracle, so it maintains information and history about the market participants.

Trader: responsible for retrieving bundles from the Oracle and trading them. Traders that fail to trade their bundle within a given number of rounds must return them to the Oracle—ensuring the market does not contain too many extra-marginal traders[1]. At anyone one time, anyone can be buyers or sellers depending on the type of the bundle received from the Oracle. Traders may also switch state—from buying to selling—which might happen, for example, if they decided that they could not acquire sufficient goods to complete a bundle's requirements given current time or budget constraints.

CDA (continuous double auction): CDAs are market-places where traders can trade a single type of resource. The seller reduces their ask-price at each round, the buyer increases their bid-price at each round and an adjudicator determines when a sale has been agreed and at what price. Traders may withdraw from a CDA at any time. An MDA is a collection of CDAs.

MDA: responsible for telling the traders which CDAs they can use to trade the resources in their bundles. The MDA stores a reference to all CDAs and if a CDA is requested for a good type which does not already exist the MDA will instantiate it.

We have run the distributed system using the same CATS input files as used in the CASS [LB03] simulations and compared the output from two sets of experiments, which has enabled us to draw a number of useful conclusions about the operation of our markets.

1.2.4 Dependencies between Goods

If Alice wishes to purchase three goods from a market, such as a flight, a hotel and an airport to hotel transfer she will likely wish to purchase all three goods simultaneously, because arguably one has no value without the other two complementary activities. However, representing the complementarities and dependencies between goods immediately makes the problem of resource allocation significantly more complex, indeed this is the problem that a Combinatorial Auction solves and we have highlighted the pros and cons of such an approach in Section 1.2.2.

[1] Buyers who have paid less than the equilibrium price and sellers who are selling for more than the equilibrium price.

With our distributed market mechanism, in which each type of good has its own continuous double auction there is no support for describing dependencies within the market mechanism. However, bundles of goods are passed by the Oracle to the Traders, whose role it is to assemble the relevant bundle of goods, through participating in multiple, but unrelated, auctions.

It is therefore very likely that a Trader will purchase the goods in series and given the limited supply of goods, they may not be able to complete the bundle. We consider the impact of bundle completion in our results (Section 6.6, page 120) as it forms one of our key measures for the success of distributed markets.

There are pros and cons to ensuring that complete sets of goods are assembled. It is a more complex problem, which increases compute time and artificially imposing a restriction to purchase a complete set may mean that consumers are denied the opportunity to purchase partial bundles of goods—there are many instances where owning some of the goods required is satisfactory. For example, Alice may find she can hire a taxi locally for less money than a pre-arranged hotel transfer, we would describe this as a complementary solution, but which is available outside of our system of markets. Whilst the dependencies between goods is an important factor and one which has been well researched, (for example in the Trading Agent Competition, described in Section 2.1.6, page 47) we do not think it is essential for systems to support dependencies between goods, there are advantages not too doing so and we evaluate "completeness" as part of this work.

1.3 Review of Potential Solutions

Given sufficient time, the CA will find the Pareto efficient solution (defined in Section 3.2.1, page 58) to the bundling problem which provides the maximum amount of revenue to the sellers/providers of goods in the auction. However, we have found (as described in Section 6.8, page 129) that a decentralised market will trade more goods with greater concurrency and solve more problems in linear time, but at the expense of finding non-optimal solutions.

We consider that non-optimal solutions are not necessarily a problem and we would characterise our solutions as "satisficing", a decision-making strategy which is dis-

cussed in detail in Section 3.2.1 and defined as [Sim55]:

> "Attempting to meet criteria for adequacy, rather than to identify an optimal solution. A satisficing strategy may often, in fact, be (near) optimal if the costs of the decision-making process itself, such as the cost of obtaining complete information, are considered in the outcome calculus."

Can we therefore solve distributed resource allocation problems, in linear time with greater predictability, using a distributed system of markets? There are a number of questions:

1. If we use a distributed market, what is the variance of the outcome between it and a centralised markets?
2. How good or bad is the result, with respect to the completeness of bundles, financial value and spend, time to complete and efficiency of the market? Are the results acceptable?
3. What affects the overlap of goods selected to complete bundles produced by the different markets?
4. Can we increase the market concurrency and/or the number of goods which are traded in the market, thus reducing social welfare deprivation?

As we have discussed there are additional fringe benefits associated with conducting resource allocation decisions in a distributed system, but ultimately, the question is can we define a menu of criteria which set out the best resource allocation mechanism for the task?

We have chosen an empirical analysis approach for two reasons:

1. An analytical approach to comparison seems infeasible — at least, given current understandings of the problem — because for complexity analysis purposes a combinatorial auction is equivalent to a weighted set-packing problem and hence can be solved analytically but it is not yet possible to perform a similar (theoretical) analysis of a system of distributed software agents — with all

the interactions and overheads that communication and distributed intelligence brings. Therefore empirical techniques offer a practical route to the evaluation of the market system and allow analysis of financial outcomes, market efficiency, bundle completion and participant satisfaction within a reasonable time frame.

2. The literature is full of examples [DJ03, WPBB01] of theoretical performance analysis that inevitably use simplified models. The use of such models then distorts any understanding of the benefit proposed by a system, particularly in the real world. Large distributed systems of autonomous objects are too complicated to model theoretically, but it is a desire to understand these systems that provides impetus and rationale for this thesis, hence we have chosen to analyse them empirically.

To carry out the comparison we identify three key attributes:

1. **Optimality**, or solution quality, whether we are maximising for buyer, seller or intermediary; are the participants getting the best solution that meets their needs?
2. **Time**, or how long it takes to produce the solutions.
3. **Cost**, or how "expensive" the approach is, in terms of items sold and bids completed, which may be evaluated through the economic surplus and social welfare produced in the system.

We will revisit these themes throughout the discussion as we build a comparison of different methods for bundling resources and evaluate their pros and cons. The approaches considered are as follows:

1. Firstly, Combinatorial Auction solver (CA), CASS, a system described and implemented by Kevin Leyton-Brown[LB03]. CAs are a type of multiple-item market and the solver takes a set of m indivisible non-identical items for sale and n bids. Each bid is a pair made up of (i) a subset of the m items and (ii) a numeric value representing the value a bidder will pay for the bundle. The CA solver will attempt to optimise the outcome of the problem such that the allocation of m items across the n bids maximises the total social welfare (sum of the buyer and seller profit) in the system.

This problem is known to be NP-complete and CA solvers search for the optimal allocation using a single, centralised auction, with complete information about all available bids, bidders and items.

2. Secondly, a Multiple Distributed Auction (MDA) is a distributed system of many single item continuous double auctions, that is many auctions take place concurrently, one for each type of good being traded. A MDA distributes the bundling operation to the "traders" in the market, who subscribe to multiple markets and buy individual items with the motive of grouping them to satisfy bundle requests.

   In doing so, the MDA shifts the system objective from achieving the maximum valuation for the sellers (the CA objective) to maximising the number of elements traded and achieving the best price for the buyer.

3. Thirdly, we re-factored the uni-processor MDA into a fully distributed and asynchronous market where Traders may buy and sell goods across multiple Continuous Double Auctions (CDAs). This was built using the AgentScape [OB06] middleware, a framework for heterogeneous, mobile agents. The use of AgentScape shifts the paradigm so that the agents are not tied to execute in sequence which enables them to act with greater autonomy increasing the competitive environment of the market.

1.3.1 Ensuring Comparability

Comparing a centralised system with a decentralised one is to some extent, an apples and oranges comparison. Both are different systems and take different approaches to solving the problem. However, in this context, they are both solving the same problem.

We took two approaches to ensuring that we had a fair basis for comparing the two systems. (i) Firstly, we used the same test data as an input to all our experiments, taking previously published distributions from the Combinatorial Auction Test Suite [LBPS00] by Leyton-Brown. This provides a large set of test data with 1000 goods and 256 bids in 500 test instances for multiple distributions spanning a wide spectrum of difficulty. Included are results from two Combinatorial Auction type solvers, CPLEX and CASS. We utilised all of the distributions in our experiments, but focus our detailed results on

the “L2” and “Scheduling” data types.

(ii) Secondly, we applied two economic tests to the systems to ensure that they produced consistent outputs. These were that the results produced are Pareto Optimal (described in Section 3.2.1) and that the mechanisms are Incentive Compatible (Section 3.2.1).

We believe that through maintaining consistency of the test environment and ensuring that all environments individually provide consistent economic properties, we have successfully provided a platform for the comparison.

1.3.2 Thesis Contribution

Typically researchers in the Combinatorial Auction space take the stance that nothing less than optimal is sufficient and look for algorithmic improvements to their heuristics and search logic. On the other side of the coin, those in Grid and Cloud Computing look to make resource allocation decisions that maximise the utility of their clusters whilst the economic properties of those allocation decisions and whether or not they could be improved upon become a secondary concern.

In deciding whether to use a centralised or decentralised system for conducting resource management activities users need to balance the trade off between utilisation, economic and social welfare of the system and compute time for the allocation process itself.

We do not re-invent the distributed market, but propose that a distributed market provides an approach for solving more difficult problems such as the job shop scheduling task simulated in our Scheduling distribution, because it shifts the responsibility for assembling a bundle of goods onto a distributed set of software agents that can grow linearly with demand, all of whom are focused on achieving maximum utilisation for their specific bundle requirement.

Uniquely, we have developed a meaningful comparison of both approaches with a view to answering the question “what is the right technique for solving my resource allocation problem” and we have conducted empirical experiments with both approaches using identical, repeatable test data..

This ensures that resources are maximised to the fullest extent possible, with good efficiency, revenue and in a predicable time frame . Combinatorial solutions will always produce a Pareto efficient option and maximise revenue for the sellers of goods, but their run times vary unpredictably based on both the algorithm used and the complexity structure of the problem, and they often sacrifice resource utilisation.

1.4 Thesis Structure

We have begun with a review of the problem, rationale and overview of the contribution that this work provides. The work is then placed in perspective in Chapter 2 (page 32), considering both the Computer Science (Section 2.1, page 33) and the Economics (Section 2.2, page 50) view. Discussion as to what the key economic properties for a distributed market might be continues in Chapter 3 (page 55), including the introduction of the game theory concepts of "Dominant Strategy", "Nash Equilibrium", "Satisficing" and "Incentive Compatibility". Finally we examine Social Welfare, a concept which can be applied to aid in decision making between multiple points on a Pareto optimal plane.

We then begin to look at our implementations. Brickworld, described in Section 3.4 (page 66) was the initial prototype, followed by the MDA system (which is used for most of our computation) and an experiment running MDA in the AgentScape environment. Although the Brickworld implementation fell short of our initial ambitions it provided a useful learning experience for the later MDA development enabling us to focus on specific areas of functionality (for example resale of surplus goods and adoption of ZI+ trading strategy, identified in Section 2.1.6, page 49).

Chapter 4 (page 74) examines the Multiple Distributed Auction system in detail, setting out the component parts (oracle, traders, auctions), the roles and responsibilities for each and gives examples to demonstrate how traders complete purchases of goods to form bundles. MDA was built using the Repast simulator and the JASA auction libraries and runs on a single machine using a stepwise approach to managing all the trader agents. It is therefore not fully synchronous and we were keen to understand what would happen if we ran the system in such a scenario. We re-factored MDA using the AgentScape framework, discussed in Section 4.6 (page 85) but found that the addition of a distributed platform meant that the large quantity of messages being

passed between Traders caused an enormously large overhead and severely impacted performance. In this section we identify a number of mechanisms to further improve MDA in a fully synchronous environment and the rationale for doing so.

Chapter 5 (page 100) looks at the problem of comparing the centralised (CASS) and a decentralised (MDA) systems and examines our approach to ensuring consistent inputs and test environments which enable a valid comparison. Importantly, we have identified a number of factors for comparison: hardness, time, financial, completion and satisfaction and we outline those in detail. We consider what tests are appropriate and the source of the test data (which comes from the Combinatorial Auction Test Suite).

To complete these experiments, we ran over 2000 test instances, each taking up to 2 hours and from the raw data we draw some conclusions in Chapter 6 (page 108). The original author of the test data we use, Kevin Leyton-Brown, published run-time and solution values for all datasets using the "CASS" algorithm which he developed, but also the "CPLEX" solver, and initially we are able to draw performance comparisons of the four sets of experiments (MDA, our CASS, KLB's CASS, KLB's CPLEX) and look at the results in terms of runtime order of magnitude. Following on from the runtime experiments we examine each dataset for each of our key attributes: hardness, financial, completion and satisfaction.

We conclude that a distributed market provides an approach for solving more difficult problems, such as the job shop scheduling task simulated in our Scheduling distribution, which ensures that resources are maximised to the fullest extent possible, with good efficiency, revenue and in a predicable time frame. Combinatorial solutions will always produce a Pareto efficient option and maximise revenue for the sellers of goods, but their run times vary unpredictably based on both the algorithm used and the complexity structure of the problem, and they often sacrifice resource utilisation.

Having concluded the experimentation, in Chapter 8 (page 149) we return to the debate—which resource allocation technique is best, centralised or decentralised—and draw conclusions? We know from the literature that many problems can be resolved optimally using centralised approaches alone and some would say there is no call for decentralised ones given the problems of ensuring an optimal solution.

In Chapter 9 (page 155) we look to the future and consider a number of scenarios in which not only the technology behind distributed resource scheduling but also the

knowledge of when to use it will be of benefit. We also examine a number of improvements for our MDA system that we feel would improve its robustness and usefulness in further work.

This thesis shows, uniquely through empirical analysis, that distributed resource allocation systems work and that whilst there are trade offs, most notably with regards the fiscal performance of the market, there are significant gains to be had, over centralised systems, in improved resource utilisation, throughput, predictable execution time and better social welfare.

Why would we want to make those trade offs? Firstly, "best technique" is a term that will be appropriate only for a specific set of circumstances and whilst this is a Computer Science thesis and therefore math related concepts such as maximising revenue might be important for some scenarios, it is important to bear in mind as we begin the journey that it is the wider economic properties of the markets and their participants which drive the need for resources. It is only within the context of human economic behaviour across distributed systems that we can truly understand the challenges that Computer Science looks to provide solutions for.

1.5 Related Publications

The following list includes all papers published by the author which are related to this dissertation. In each case my contribution to the paper is stated in accordance with regulation 16.1 subsection 3.v of the University of Bath regulations.

[GOT+08] Engineering Large-scale Distributed Auctions (Short Paper) Gradwell, P.; Oey, M. A.; Timmer, R. J.; Brazier, F. M. T. and Padget, J. Proceedings of the Seventh Int. Conference on Autonomous Agents and Multiagent Systems (AAMAS), ACM 2008

We believed that the functional characteristics of market-based solutions are typically best observed through the medium of simulation, data-gathering and subsequent visualisation. We previously developed a simulation of multiple distributed auctions to handle resource allocation (in fact, bundles of unspecified goods) and in this paper we wanted to deploy an equivalent system as a dis-

tributed application. We worked with the AgentScape platform and colleagues in the AgentScape development team at VU University, Amsterdam.

There are two notable problems with the simulation-first, application-second approach: (i) the simulation cannot reasonably take account of network effects, and (ii) how to recreate in a distributed application the characteristics demonstrated by the mechanism in the simulation. We describe: (i) the refactoring employed in the process of transforming a uni-processor lock-step simulation into a multiprocessor asynchronous system, (ii) some preliminary performance indicators, and (iii) some reflections on our experience which may be useful in building MAS in general.

The AgentScape reimplementation work described in the paper was completed by Michel Oey, Reinier Timmer and myself. My focus was on the MDA design and implementation whilst Michel and Renier focused on replacing the Repast engine with the AgentScape framework. The paper was jointly written under the supervision of Francis Brazier and Julian Padget.

[GP07] A comparison of distributed and centralised agent based bundling systems Gradwell, P. and Padget, J. ICEC '07: Proceedings of the ninth international conference on Electronic commerce, ACM Press, 2007, pages 25-34

In our ICEC paper we argued that the use of trading agents to manage the allocation and bundling of resources across computer networks is well established and literature to date has focused on a variety of auction and distributed market type mechanisms that use economic principles to determine the "best" allocation.

Having conducted an early empirical analysis of a number of solver algorithms, principally the Centralised Combinatorial Auction Solver (CASS), we had shown that those using bounded search techniques are typically able to solve a majority of cases in linear time, while there remain a number of outlier cases that are computationally problematic. In contrast, distributed mechanisms are intrinsically less than optimal for sellers, but demonstrate significantly less variance in computation time.

A proper understanding of the different performance properties and suitability of the different techniques is necessary in order to make an informed choice between a distributed market and a centralised auction. Consequently, we have completed an empirical evaluation of CASS, a centralised mechanism, against two distributed mechanisms: (i) Multiple Distributed Auctions (MDAs) and (ii) Quote Driven Markets (QDMs). Uniquely, we carry out simulations of all three mechanisms using a common dataset, generated by the Combinatorial Auction

Test Suite (CATS), providing a real basis for comparison. The main results presented are that distributed mechanisms deliver (i) increases in the number of items traded (ii) a greater proportion of bidder requirements being satisfied, but (iii) potentially less optimal bundle solutions and (iv) consistent run times with low overall variance when compared with centralised algorithms.

This paper contained the first results from our CASS and centralised MDA experiments and later versions of these experiments form the bulk of results presented in this thesis.

The work described in the paper was completed by myself and the paper was edited by Julian Padget.

[GP05] Markets vs auctions: Approaches to distributed combinatorial resource scheduling Gradwell, P. and Padget, J. In journal "Multiagent and Grid Systems", 2005, 251-262

In this initial paper we introduced the concept of Grid computations and argued that they will be enabled by participants trading resources in order to construct bundles of goods or services that constitute experiments in science, engineering and now emerging, social sciences. A combinatorial auction (CA) is a natural choice for optimal resource allocation, but the space and time dimensions that characterise a Grid would appear to indicate they are incompatible. This paper proposes that an analogue of a physical commodities market seems more appropriate and that there is a class of bundling problem whose complexity properties appear to make the utilisation of a CA impractical.

We describe our simulation environment, BrickWorld, which comprises a distributed tier of "TraderAgents" and multiple distributed single item auctions (MDAs). The issues associated with the complexity of bundling are evaluated, in particular those arising when attempting to provide useful combinations of items in situations when the multi-dimensionality of the bundle would make it impractical to finish the NP-complete optimisation successfully in the soft real-time setting that is the Grid.

Finally, the evaluation strategy presented helps demonstrate that for small bundling problems, a single CA continues to provide a high level of performance, but as the complexity level of the problem increases and the problem becomes distributed a system of MDAs may prove more effective.

The work described in the paper was completed by myself and the paper was edited by Julian Padget.

Chapter 2

Review of Literature

Computational markets and associated topics are studied and discussed across both the Computer Science and Economics literature and this thesis draws comparisons across that range, specifically evaluating the problem in light of both Computer Science - Grids, Agents, Distributed Computing and Resource Management and Economics - Pareto Efficiency, Social Choice, Social Welfare and Market Based Control. Multi-Agent Resource Allocation (MARA) is one name given to research which spans the work of Computer Sciences' Intelligent Agents and the Economics topics of mechanism design and game theory. In a comprehensive survey paper on the subject [CDE+06], Chevaleyre et. al. suggest that Computer scientists often take the procedural view "how do we find an allocation?", whilst economists are more likely to concentrate on the qualitative "what makes a good allocation?".

The literature is wide ranging and in order to understand the state of the art in resource allocation and markets we need to consider work on topics including: (i) Preferences, (ii) Social Welfare, (iii) Complexity, (iv) Negotiation (v) Algorithm Design, (vi) Mechanism Design, (vii) Implementation (viii) Simulation and Experimentation and (ix) the interplay of theory and applications.

In this review we found that there is a large body of work which describes both the design and attempted implementations of market simulations, usually with a specific focus. Frustratingly, there is little concrete discussion as to how these simulations were assembled, or what their actual performance was and therefore it is subsequently impossible to build comparisons or benchmarks of either their system or ours.

Secondly, authors appear to fall into one of two camps, those for centralised markets and those for decentralise and there does not appear to be a body of work devoted to comparing and contrasting the two viewpoints, which is a void we seek to address in this thesis.

2.1 The Computer Science View

It is important to understand the current state of the art so that this thesis may be placed in context and the gaps in research identified. Taking the Computer Science perspective, we have chosen to group the literature into (i) work done on market simulations, (ii) discussions on centralised and (iii) decentralised approaches to resource management, (iv) market design and (v) the Trading Agent Competition; this last because it serves to illustrate a number of plausible solutions for distributed resource allocation.

Empirical modelling of markets has proven to be a difficult task, due to the complexity of defining the rules for the market, the number of parties involved and the volume of data and trading that occurs. Attempting controlled scientific experiments in these areas is very challenging and Market Simulation literature shows us that a number of different approaches have been taken.

Within the market modelling community there is a divide between the centralised and decentralised proponents. The former is characterised by an optimal, but NP-Hard solution whilst the latter offers a more pragmatic approach with useful (although not optimal) results in linear time. It is necessary for us to understand the algorithmic state of the art in both sectors so that we can demonstrate how our system compares.

Finally, we consider the Trading Agent Competition (TAC) as it has provided the major forum for empirical research into agent and market strategies.

2.1.1 Distributed Market Simulation Engines

Apart from TAC and its associated toolkits there are few documented simulations of distributed market systems. Nimrod [ASGH95], a tool for building a specialised para-

metric modelling system using distributed workstations is one such system that incorporates a distributed scheduling component. The system is interesting in that it can manage the scheduling of individual experiments to idle computers in a local area network. The Nimrod scheduler has been used for modelling power grids, drug design and computer networks. However, Nimrod has not been subjected to any performance comparisons or analysis of the relative benefits of the system when compared to others.

Looking specifically for software agent driven approaches we are interested in Cao's explanation of his PhD thesis, an Agent-based Resource Management System for grid computing (ARMS) in [CJS+02b, CKN01]. ARMS follows the traditional pattern of using agents to provide service advertisement and discovery for application scheduling on the Grid. One of the interesting aspects of Cao's work is the use of a toolkit called "PACE" [NKP+00, CJS+02a]. The PACE environment provides a quantitatively based evaluation of a distributed systems performance but it is focused on the evaluation of parallel systems and supercomputers (e.g. those using high-speed message-passing-interfaces for communication). Cao's simulation was completed on a cluster of nine multi-processor Sun Ultra 1 workstations. The interesting highlight of this work is that optimal run time performance appears to be obtained at approximately 16 processors, although the reason for this is not discussed in detail. At a superficial level, this is similar to our experiences with the AgentScape Platform [GOT+08].

2.1.2 Centralised Approaches

The objective of a resource allocation procedure is either (i) to find an allocation that is feasible (e.g. to find any allocation of tasks to production units such that all tasks will be completed in time) or (ii) to find an allocation that is optimal.

As a good introduction Biswas describes the detail of an iterative Dutch combinatorial auction in [BN05]. This paper gives some useful background on how CAs may be modelled as a weighted set-packing problem and uses generalised Vickrey auctions, as described in [CSS05, Ch.1], to derive worst-case bounds for the algorithms. This paper is helpful in shaping our understanding of the structure of a CA and its worst-case performance.

Like other authors in papers such as [SCG07], Biswas supports the notion that the primary measure of agent performance should be total profit over a simulated period

of activity. This is an important theme we develop later when considering how to measure our distributed markets.

Sandholm has authored a great many papers on different aspects of centralised approaches to scheduling and resource allocation problems and much of his research builds on real world experience gained through CombineNet, a company founded to exploit his various algorithmic patents. Unfortunately the commercialisation of his work has meant that there is little detail of its practical applications. However, in his keynote talk at the International Conference on Electronic Commerce in 2007 [San07] he outlined how they have developed a new paradigm called "expressive commerce" and applied it to industrial procurement. The concept is that they capture a huge amount of heuristic information from buyers and sellers so that buy and sell requirements are much more expressively defined using a wide range of quantitative factors than in previous attempts. Buy and sell requirements are algorithmically cleared using a centralised combinatorial auction—but in reasonable time frames, with qualitative factors used to prune the results. A number of developments have lead to this point. Early work on the CABOB algorithm [SSGL01, SSGL05] demonstrated a mechanism for applying pruning heuristics to give an algorithm which could solve most problems in polynomial time. Subsequently Sandholm published [San02] an "Algorithm for optimal winner determination in combinatorial auctions" which refined the work by: (i) provably finding the optimal solution, (ii) searching a graph whilst completely avoiding loops and redundant generation of vertices's, and (iii) took advantage of the sparseness of bids by focusing on tree-segments which are most likely to be relevant.

These strategies meant that they produced an algorithm with provably polynomial run times that is arguably still the pre-eminent approach to solving resource allocation problems through a centralised decision process. However, since the system is commercialised we have been unable to perform any quantitative comparisons of our work against it, although others have demonstrated algorithms which are both provably optimal and which operate within a bound of optimality, such as Dang [DJ03] who presented new clearing algorithms for multi-unit single-item and multi-unit combinatorial auctions with piecewise linear demand/supply functions. Following a complexity analysis of the algorithms (where the complexity function of the algorithm is $O(n - (K + 1)^n)$) where K is the upper bound on the number of segments of price available) the authors were able to prove that they are guaranteed to find the optimal allocation.

Finally, Sandholm has always taken the view, reiterated in [San02, San06] that the approach of compromising optimality to achieve a polynomial time solution is futile because one may end up with worst-case approximations and in many situations those would represent disastrous outcomes for the participators in the market (for example, a Government might auction radio spectrum at a massive discount if they did not provably have the optimal outcome, thus denying tax payers of significant revenue streams and leaving the door open for allegations of corruption and incompetence). However, we would argue that whilst there are some scenarios where this is indeed the case, there are others for which the need to determine a solution to a complicated problem will quickly outweigh the benefits of achieving the optimal solution some time in the future.

Our view is echoed in work by Wolski, Plank et al [WPBB01] in their paper "Analyzing Market-Based Resource Allocation Strategies for the Computational Grid" where they compare market strategies in terms of price stability, market equilibrium, consumer efficiency and producer efficiency and find that using a commodities market for controlling Grid resources gives greater price stability, market equilibrium, consumer efficiency and producer efficiency than previous auction based approaches. They implement a number of simulations using the following constraints, which differentiate their work from our MDA simulations: [WPBB01]

1. All entities except the market-maker act individually in their respective self-interests.
2. Producers consider long-term profit and past performance when deciding to sell.
3. Consumers are given periodic budget replenishment and spend opportunistically.
4. Consumers introduce workloads in bulk at the beginning of each simulated day and randomly throughout the day.

Having implemented the theoretical work of Smale [Sma76] as well as their own "Bank of G" strategy they found that an auction was inferior for resource allocation for all their simulations and that a commodities market offered better price stability and equilibrium as well as increased resource utilisation (for example, the average percentage of time that a CPU resource on the grid is occupied is higher under the distributed mechanism). Whilst the constraints they have applied to the simulation make the work

different to MDA, it is useful to note that others have formed similar conclusions to us with regards the best way to conduct distributed resource allocation.

Parkes [Par03] has also surveyed the problem of distributed bundling and how it could be achieved with a multi-agent system, suggesting that the best way forward is to tackle the combinatorial complexity problem by distributing the processing load imposed by the issue across a network of agents. Rather than attempting to perfect a solution to the full NP-complete problem, the network solution would provide as good an approximation as could be developed in the amount of time available for completing the search of the problem space, in effect an anytime approach, and utilising that outcome would give the user a "best-efforts" allocation of resources which would have greater utility for them than no allocation at all.

2.1.3 Rational Choice and Market Design

Our work uses software agents, which are computer programs that emulate participants in a trading market with objectives to buy and sell bundles of goods. In order to complete these trades, the agents must make decisions, hold preferences and follow a strategy to ensure that they complete sufficient trades at the right price so that they successfully deliver complete bundles. The strategic behaviour of the software is modelled on human, social and economic behaviours and the development of formal models of this behaviour is called Rational Choice theory.

If Rational Choice theory is the modelling and development of the Agents, then "Market Design" or "Mechanism Design" is describing research into the design of market strategies and the preference elicitation of participants in those markets. Research in these two areas of work provides the foundation for the different mechanisms we consider in this thesis.

Herbert Simon introduces the concept of rational choice in his paper on Satisficing [Sim55] and Shneidman and Parkes have co-authored a number of papers [SP03, SP04] discussing rational choice in the context of mechanism design in networks of peer to peer devices. Rational choice is the basis of fair decision making and it is an issue that occurs in theoretical economics, practical market design and grid resource scheduling. Shneidman and Parkes' concern is that agents in a peer to peer trading network are not likely to act rationally (they may gain some advantage by not doing so) and therefore

we need to incentivise them to follow protocols that provide the system as a whole with good performance. In [SP03] they put forward a number of principles from Mechanism Design as a methodology to use when designing the protocols.

The concept of nudge economics is related, and the subject became popular with the publication in April 2008 of a book "Nudge" [TS08] by Richard Thaler and Cass Sunstein. The concept behind Nudge Economics is that humans are not inherently rational beings and sometimes need a little guidance—gentle nudges—in order to make the best choices in a given situation or behave in the most appropriate way. From an economics standpoint what they say about human nature is controversial because this brand of economic thinking sometimes referred to as "behavioural" or "new" economics goes against more neoclassical economics and the economic man theory which claims that individuals always act rationally. It is our view that humans do make rational decisions (although occasionally they make a rational decision about an irrational subject) and that therefore, because software agents might not, it is important that our market encourages (or nudges) rational behaviour.

In the context of future Grid network technology a rational choice is one of the problems faced in developing efficient mechanisms for provisioning services to clients by a scalable and dynamic resource allocation (matching) mechanism. The CATNETS project considered this in their evaluation of the Catallaxy paradigm for decentralised operation of dynamic application networks. Catallactics is the science behind self-organising free market systems and the way in which there is an exchange of values and price negotiation and a catallaxy is a self-organising market mechanism originally described by Friedrich August von Hayek [vH76]. A catallaxy differs from a traditional "economy" in that the agents in the system will not have shared goals. Eymann et al [ERS$^+$05] outlined the Catallactic approach to resource allocation: Firstly, a service market in which complex services are traded on the basis of price and availability, whilst secondly, there is an allocation layer which assigns the purchased resources to specific jobs. The paper provides a helpful description of other work undertaken in the area of distributed markets, but the major contribution is to introduce this concept of a Catallaxy into Grid computing.

The different treatments of rational choice across computational economics show that in designing markets one needs to be careful to ensure that we do not encourage negative behaviour when building our own markets.

2.1.4 Implementation of Auctions and Markets

We have discussed previously that it is difficult to build empirical simulations of markets and the following papers provide examples of previous implementations, identifying some of the pitfalls and successes for us to be aware of. In addition, we have identified a number of desirable properties for our MDA market: (i) Mobility of agents and (ii) fair and stable pricing, and we can look to these example implementations and analytic tools to discover best practise.

The SPAWN system [WHH+92] is the earliest known market-based computational system which utilised idle computing resources in a distributed network of computers that showed that high communication overheads lead to unstable market pricing, a scenario we should seek to avoid. The objective was to increase utilisation of computer processors, though a set of sealed-bid, second-price auctions but the project discovered that whilst their mechanism scaled there were price fluctuations across the markets as the number of participants grew, due to the limited communication of information between SPAWN systems, which were configured only to talk to their neighbour and not in a full mesh. The authors recommend that this be investigated in further work and indeed this is something we report on later as we have been able to investigate the impact of communication between a large number of trading agents in our MDA-AgentScape work. SPAWN identified two further deficiencies: (i) It existed as a user-space C application and was therefore unable to offer security to the host operating system or the ability for processes to migrate between machines, (ii) it did not provide applications with robust recovery in the event of failure (e.g. due to machine failure or early termination of jobs due to insufficient resources).

Bredin tackled the problem of mobility [BKR98]—where agents move from one machine to another—and described a system in which mobile agents purchase resource access rights from host machines thereby establishing a market for computational resources. The use of a market mechanism allowed precise communication about the quantitative utilisation of resources which gives a metric to the agents allowing them to distribute themselves evenly throughout the network. Through simulation they showed that it was possible for resource suppliers to react to agent demand and calculate an optimal pricing strategy and hence a profitable allocation. Similarly, agents were able to plan their expenditure and maximise their utility.

Chaun and Culler introduced [CC02] the concept of a performance analysis for market-

based batched scheduling of jobs on grids using clusters of commodity work stations in 2002. Their modelling relied on "user-centric" performance metrics as their basis for system evaluation with each user having a utility function which measured value delivered as a function of execution time. Chaun and Culler define their utility function using two methods. Firstly, they ask all users to assign a utility value to each job using a common medium of expression (they use currency, which is convenient for a market based system). Secondly, they assume that a valuation is a piecewise-linear function which will decay linearly over time after the predicted completion time has passed (i.e.: the users value remains constant until the amount of time that their job would have taken to complete on dedicated resources has passed). The sum of these two functions then defines the user's utility and is used as part of an aggregate utility calculation to quantify overall value delivered to end users. Through the use of modified scheduling software which used end-user value as their utility function Chaun and Culler were able to observe performance improvements of 2-5x for sequential workloads and up to 14x for highly parallel workloads. Their work shows us that understanding end-user utility enables us to make more informed resource allocation decisions and increase throughput in our market mechanisms.

There are two further papers of note covering this topic which add weight to the notion that empirical analysis of markets is feasible. Firstly, RECO: "Representation and Evaluation of Configurable Offers" [BKL03] describes a mechanism for representing bids using propositional logic and then a decision support tool helping the buyer to procure an optimal configuration for a single good. Its limitation appears to be that it is restricted to the context of one buyer and multiple sellers, for a single good. Secondly, Rachel Bourne's 2003 paper [BBP] described the need for intermediaries in continuous markets. This provided the inspiration for our Quote Driven Market (QDM) model and our subsequent empirical analysis of the QDM.

The work done by Bredin, Chaun and Culler and Bourne all give us a strong basis for our assertion that an empirical approach to resource allocation with the objective of producing a "statisficing" outcome is a viable and tested objective.

Our distributed market was initially implemented both in the Repast [NCV06] simulation engine and subsequently in AgentScape [OB06], a distributed agent simulation environment. Repast is a free and open source agent-based modelling toolkit that offers users the ability to create their agents as independent Java objects which will then execute for a number of steps until completion. We used Repast as it provided an easy to

use simulation environment and allowed detailed step by step analysis and debugging of our distributed market system. AgentScape is a full agent platform, providing the kernel, security, mobility and message passing environment needed to excecute agents across a number of distributed nodes. Our implementation of a distributed auction was the first for AgentScape and we discuss our refactoring of the Repast implementation into AgentScape in Section 4.6. There are alternative agent platforms which could have been used to support our development, including JADE [BPR99] and Cougaar [MT04]. Primarily we selected AgentScape for it's truely distributed nature but practical factors, including existing relationships and availablity of technical support played guiding roles. We also found a number of indepth evaluations of different agent platforms very useful, including work by Railsback [RLJ06] and an evaluation of the freely available java simulation libraries by Tobias [TH04], which highlighted the Repast engine as a high performing environment, ideally matching our objectives for running simulations with large quantities of traders and auctions. Finally, a more recent survey by Nikolai [NM09] provides an updated (2009) overview of agent platforms.

Security and Integrity in Markets

Markets are susceptible to a number of attacks and interruptions, both malicious, such as false name bidding (impersonating other bidders) and unintentional, e.g. corruption of bid/shout messages during transmission between participants. In our MDA work, we have sought not to focus on security because this is an area that has been comprehensively researched and simply ensuring our markets are secure systems does not add useful benefit to our comparison. However, it is helpful to know that in the event of production usage, we could implement the techniques considered below:

Transactional Integrity: [FR95] Franklin and Reiter have designed a secure auction service, which is of merit because it provides a distributed platform for running sealed-bid auctions. The secure function of their market comes from the development of a novel cryptographic technique that can ensure that (i) the bids of correct bidders are not revealed until after the bidding period has ended, (ii) the auction house collects payment for the winning bid (iii) losing bidders forfeit no money and (iv) only the winning bidder can collect the item they bid upon. However, they do not give much detail on their auction mechanism or its distributed properties and as a result this paper is only useful to state that such security mechanisms could be integrated if required.

Identification of parties and protection against impersonation: In this early paper [SS99] from 1999 Stubblebine and Syverson considered the problem of impersonation attacks on on-line auctions and the affect these have on fair negotiation. Using a process of notaries and a certified delivery service (but in this case there is no need to trust the auctioneer) they are able to present an on-line English auction in which bids are processed and the auction is cleared fairly.

Secure Message Passing: If one was to consider developing a message passing system, there are a number of plausible solutions for dealing with issues of message corruption, fraud, interception. There are three obvious approaches: (i) Lamport's "Byzantine Generals" Paper [LSP82], (ii) Leader election style [PB99, RNSP97], and (iii) Chaining solutions [SP04, SP03].

The Byzantine Generals' problem defends against Byzantine failure in which a component in the market would not only behave erronously, but also inconsistently. Fault tolerance in this scenario is ensured by duplicating their messages and ensuring that the different agents participating in the market execute an unanimous decision.

In an agent based system a leader election is the process of designating a single agent as the organiser of a distributed task. The algorithm was invented for managing the control token in a token ring network, where all nodes need to communicate between themselves in order to decide which is the "leader". The agents use a domain—specific algorithm to break the symmetry among identical nodes.

Shneidman has done work on the rationality of self interested agents, which creates a potential problem of message passing:

> Imagine running an auction over a large peer to peer network. You are the auctioneer and have three directly connected neighbours. You send out an announcement advertising the auction and ask that it be globally propagated. You then sit back, expecting many bids, and are surprised when you only receive three bids one from each neighbour.[SP04]

In this peer to peer scenario, Shneidman defines a mechanism where it is beneficial for the agents to pass the message onwards to their peers in such away that it continues to be passed from one agent to the next, until all agents have participated. The mechanism should be designed such that there is no gain from cheating and so that cheating can be detected where possible, with the use of

sanctions in the event that members do cheat. Parkes has developed this work [PS04] to ensure that messages are not corrupted in transmission by proposing a message passing structure, in which each agent passes its bid to the next agent, along a chain (or tree), with the agent at each step determining the current winning bid by comparing the bid they receive with their own until the result reaches the seller. This mechanism also has the benefit of linearly distributing the processing overhead of the allocation decision across multiple agents so that NP-Hard computations are replaced with a set of sorting functions, which mitigates one of the further problems seen with combinatorial auctions.

2.1.5 Grid Resource Management

The aim in Grid Resource Management is to examine how multi-agent systems may be used to develop a market place for the trading of computational grid resources such that grid scheduling is completed without the limitations of the top down approach (direct, centralised control, complete information, etc).

Newly available cloud computing infrastructures, such as those by Amazon, are beginning to make accessible huge amounts of computing power on a "just in time" basis. However, the economic models surrounding these systems are static and uniform, with charging models that, for web-based cloud systems work on a price per unit per hour basis. Is there work being done to look at the linkage between pricing and resource availability?

All grid scheduling systems work on the basis that the new task to be executed has to make itself known to a resource selector. In current systems, the resource selector acts as a gateway to the grid. It will select resources from a global directory (e.g. the Globus MetaDirectory Service[Pro]) and then allocate the job to one of the available grid nodes. Typically, job allocation is done in two stages. Firstly a job is allocated to a particular node on the grid and then within that node, the job will be scheduled onto a processor.

Typically in grid scheduling literature the first practise is called resource allocation and the second is job scheduling. Both approaches have been investigated in the literature, under the term "meta scheduling", for example in the AppLeS project, discussed in [BW96] and [BW97]. AppLeS looked at "application-centric" scheduling, a con-

cept born out of the needs of users of modern distributed Grids, where the use of fast networks to aggregate distributed CPU, memory, storage and data meant that a large meta-computer could be used, but equally, users were finding it difficult to harness that power because of conflicting end-user requirements. In the AppLeS system, each user of the cluster has an intelligent Application Level Scheduler (AppLeS) agent which implements an application specific schedule. This promotes the performance of the application by evaluating every resource allocation decision in terms of its impact on the application's resource requirement. The AppLeS project is now finished, but Berman continued the research themes with the GrADS project [BCC+01] provided a Grid Application Development Software set, a set of software libraries to support the execution, monitoring and development of Grid aware applications. They developed a number of tools, including a simulator "MicroGrid" which allowed the emulation of a Grid computing environment.

Buyya made a major contribution in his thesis [Buy02] by introducing economic techniques, outlining the concept of a Grid Architecture for a Computational Economy (GRACE). This work also produced GridSim, a simulator for global Grid environments, Libra [SAL+04], an economy based job scheduling plug-in for GridBus and the GridBus project which fosters research on Grid management tools and applications of Grids in areas of e-Science and a taxonomy of market-based resource management systems [YB06] in which he comprehensively considers all of the available resource management systems for economy based Grid computing systems including the distributed, management control domain. His work helps us understand the deficiencies in the traditional models of grid resource management (which we seek to correct) that are best understood through public sector grid computing systems, which typically operate on a zero settlement basis. Users give their resources for free and researchers connect across their networks to utilise them. Within a closed user-group, for example a research facility's own "mini-grid" of computing resources, scheduling policies may differ but they are typically still managed by a system (e.g: Globus, GridBus) that executes jobs on a first-come, first-served basis or which prioritises jobs according to local circumstances. Consequently, availability of computational resources is systematically unpredictable and the various methods employed, outlined in Buyya's description of Libra [SAL+04] (an economy based grid scheduler) such as first-come, first-served, time slot, priority or availability-based scheduling of jobs on a grid cluster all operate in a top-down manner and require knowledge and direct control of all the jobs on the grid.

A system in which there are multiple entities all attempting to acquire part of one entity is always going to experience concurrency problems. According to [VD02], the meta scheduler architecture devised in the GrADS system [BCC+01] suffers from a deficiency in that if two jobs are submitted to the grid at the same time then they will both be processed without regard to whether the demand can be met. Leinberger et al., proposed a solution [LKK99] but observed that the approach is an inefficient scheduling method for the type of computing infrastructure in which clusters have multiple types of independently allocatable resources such as shared memory, large disk farms and distinct I/O channels. Many schedulers, designed for use in single resource pool environments use a technique known as "back-filling" to select jobs from farther down the queue for immediate execution, resulting in a lower average response time for smaller jobs whilst guaranteeing a level of progress to larger jobs. However, back-filling is still subject to a phenomenon known as resource depletion when used in environment with multiple resource types because back-fill methods typically use a greedy first-fit criteria in job selection. This can result in a scenario of jobs that have consecutive high levels of requirement for specific resources, leaving other types of resource under-utilised. The essence of Leinberger et al's argument [LKK99] is that by developing the back-filling algorithm so that it becomes aware of the multiple different pools of resources, such that it keeps their utilisation balanced then it is likely that more jobs will fit into the system overall as there will be no need to delay jobs if a specific resource pool is over-utilised. Leinberger showed that as the number of resource pools increased their "back-filled-balanced" heuristic maintained significant performance gains (in general outperforming by up to 50%) over a simple back-filling approach.

This shows us that when building a resource allocation model for distributed markets, which by their nature contain a number of pools of different resources, it will be important for us to ensure that all of the pools are as equally balanced as possible to mitigate against uneven resource demands. This collision problem, of goods potentially being sold twice, remains a characteristic of a trading based resource allocation system. A trader might deliberately oversell or under-price. The problem of double-booking at the resource allocation level should be dealt with by market forces, that is Traders should be able to oversell, but, if they do so and their clients suffer they will rate the trader badly and therefore the trader would not do so well next time, as happens with airline seat reservations.

2.1.6 Methods for distributed resource allocation

When faced with a specific allocation problem it is helpful to be able to determine the most appropriate method to be used in order to compute the optimal allocation. Is a financially driven auction the best choice or should a trading market be used? Distributed resource allocation is not a problem limited to grid computing networks and there is much literature on the subject outside the Grid computing sphere particularly with respect to distributed algorithms.

In their seminal 1980's paper, "A Micro-economic Approach to Optimal Resource Allocation in Distributed Computer Systems" Kurose and Simha [KS89] consider the benefits of using an economy-based decentralised algorithm for file system allocation.

Kurose and Simha argue that there are two possible classes of resource allocation mechanisms: price-directed and resource-directed. Many existing resource allocations operate using a resource-directed approach. Whilst, in a closed and trusted system this may work effectively, it is not suitable for an open competitive environment. This is because, when a system is dependent on all parties developing a common valuation scale all valuations rely on trust and cooperation between all the participants seeking resources. This is in contrast to finance-based systems which can perhaps be more easily measured and are more widely understood.

Price is the ideal valuation function for this setting for two reasons: (i) price can ultimately be seen as a projection function from a multi-dimensional vector of "values" — both quantitative and qualitative — to a single value and (ii) secondly, in a free market, over the longer term the price, cost and the marginal value that a user assigns to a good will all converge as competition drives down price and makes a number of good alternatives available.

Work on similar problems in other related areas (parallel computing, file system scheduling, etc.) [ASGH95, FBK96, SH80] typically seems to focus on a simplified environmental model. A key weakness is that it lacks the free market or "real world" aggressiveness and competition which a monetary based allocation environment (auction, trading floor, etc.) should seek to emulate.

Trading Agent Competition

The Trading Agent Competition succeeds in illustrating the essence of the distributed resource allocation problem as it deals with distributed resource bundling through auctions. The competition has run in various guises since 2002 and has provided a forum for the development and discussion of many trading agent scenarios. There are presently three games:

TAC Classic: The Classic game is a "travel agent" type scenario based on a complex procurement requirement of flights, hotels and entertainment events.

TAC SCM: The Supply Chain Management game is a Personal Computer (PC) manufacturer scenario based upon the sourcing of components, manufacturing of PCs and sales to customers.

TAC Market Design: The Market Design game (or CAT) is a reverse of the classic game. The software trading agents are created by the organisers of the competition, and entrants compete by defining rules for matching buyers and sellers and setting commission fees for providing this service. Entrants succeed by attracting buyers, sellers and making profits.

The classic Trading-Agent Competition (TAC Classic) and the supply chain scenario (TAC SCM) were motivated by the desire to develop automated strategies for buyer and seller software agents in marketplaces. The trading rules or interaction mechanisms are fixed by the TAC Classic/TAC SCM organisers, and competition entrants compete with one another by creating agents that seek to trade under these fixed rules. As mentioned, CAT is the opposite game.

In addition, TAGA [ZFD+03b, ZFD+03a] is a redevelopment of TAC by the Agent lab at UMBC and provided the initial inspiration for our MDA system. The key advance of TAGA over TAC was that the TAGA team opened up the TAC interface and made it a distributed agent based system which uses web services for communication.

TAC provides useful lessons in risk management because the travel agents must buy the various components: hotels, flights and entertainment in separate auctions using separate strategies for each. This creates the risk that they may for example acquire

a hotel, but not the corresponding flight, in which case the agent will fail. Significant work has been done on various strategies to cope with this uncertainty with the SouthamptonTAC [HJ02] agent achieving the highest mean score and lowest standard deviation in the Second International Trading Agent Competition (across 600 games). In that paper the authors highlight those strengths which they perceive to have been key. In the TAC competition, users purchase sets of flights, hotels and entertainment activities and due to the uncertainty and unpredictability of resource availability, there is no optimal strategy appropriate for all situations. For this reason, the SouthamptonTAC agent works in "rounds" and re-calibrates the optimal distribution of goods to customers for each round. It uses learning techniques to determine subsequent bidding actions and continues in a loop until all the rounds have been completed. One of the interesting factors in the TAC competition was the approach to risk adopted by the various agents. The authors found that risk-seeking agents would buy lots of flight tickets early on and rarely changed their customers travel plans, which then meant that they performed badly when prices of subsequent activities (e.g. hotels) were high. In contrast, risk-averse agents do not bid, but are similarly unable to meet their customers travel plan requirements.

SouthamptonTAC is broadly risk adverse and but can adapt its behaviour to take more risks when it finds that games are not competitive and prices are fluctuating too much. The key point from this learning is that when dealing in an agent-driven market the authors argue that it is not feasible for the agent to have a single strategy and that they must adapt through the life of the market.

TAC-SCM has also provided useful learning outcomes. Ketter et al [KCG$^+$07b] present a predictive empirical model for pricing and resource allocation decisions. They argue that by improving the price prediction system we can build better decision support systems which act rationally on behalf of their users and their approach was based around a learning approach from previous work using a Markov prediction process[KCG$^+$07a], predicting market prices and price trends and estimating the probability of receiving an order for a given offer price. This work, as demonstrated by its effectiveness in the TAC-SCM games, is helpful to us in determining potential bidding strategies.

One of the problems with empirical experiments in agent communities is that of evaluating them. Market simulations, like the real-world are typically highly complex, variable and traders frequently have incomplete information. This gives rise to a very

large potential set of strategies and market conditions, hence making empirical analysis very difficult. In [SCG07] Sodomka, Collins and Gini propose a mechanism for improving the statistical analysis of market simulations by controlling their complexity and allowing simulations to be run with a number of controlled variables. One of the benefits of the TAC-SCM project is that all the different participant's implementations are made available and so it has been possible to re-run the experiments under controlled conditions. From this work they showed that they could perform statistical significance testing with fewer games than previously possible and the additional market control also enabled them to look at interactions in the game which previously had been masked by variability and noise in the markets.

The team at Southampton have also been investigating a number of different bidding strategies in order to support their work in the various TAC projects and Anthony also looked at online auctions [AHDJ01, AJ03] focusing on how to make an agent which could participate in multiple auctions making purchases on behalf of a consumer. Similarly to Sodomka, this paper discusses how they ran their agent in a simulated environment and gave an empirical evaluation of its performance.

Zero Intelligence Trading

Most of the work in TAC Classic and TAC SCM focuses on having intelligence in the traders who are performing the buying and selling of resources. However, in his 1997 technical report [Cli97, CB98], Dave Cliff develops the concept of "Zero Intelligence Agents", initially introduced by Gode and Sunder. The ZI Agent is one that acts randomly within a structured market and Cliff uniquely demonstrated that average transaction prices of ZI traders can vary significantly from the theoretical equilibrium level when supply and demand are asymmetric, and that the degree of difference from equilibrium is predictable from a priori statistical analysis. Cliff follows by introducing "Zero-Intelligence-Plus" agents, whose performance is significantly closer to that of humans than simple ZI traders could achieve.

2.2 Economics view

We need to draw from the economics literature on themes such as Pareto Efficiency, Social Choice, Social Welfare and Market Based Control to fully understand the behavioural foundations of computational markets and to avoid reinventing the wheel. These themes identify both important behaviours that we should seek to encourage in our markets and justifications for those less desirable behaviours that we may observe. By briefly understanding the origins of a few pure economic concepts we can have a more relevant understanding of Computational Economics which will assist us in our later market design discussions.

Social Welfare was introduced by Bergson [Ber38] in a 1938 paper on Welfare Economics, which outlined the principles and gave a precise definition of the conditions required to engineer the maximum economic welfare for a given scenario. The Mara Survey [CDE+06] says that there are many different notions of social welfare. In the context of an ecommerce application the aim may be to maximise the average profit generated by the negotiating agents. However, if agents are trying to agree on something less money-orientated, such as access to an Earth Observation satellite which was jointly funded by its owners then the priority might be to ensure that each agents gets their fair share of the common resource. We discuss Social Welfare in more detail in Section 3.2.1 (61).

In 1950 Arrow developed a theorem which enables us to characterise social welfare and gives a number of criteria which essentially shows that it is impossible to please everybody. This "impossibility theorem" was introduced in [Arr50] and Arrow developed this work to look at what happens if you have an economy characterised by the presence of asymmetric information. In 1971 he showed [AH71] that you cannot expect a competitive equilibrium unless you have symmetric information, giving us a formal economic basis for the rationale behind requiring complete information in a centralised decision making process.

Herbert Simon [Sim55] introduced the concept of "satisficing", a decision-making strategy which attempts to meet criteria for adequacy rather than to identify an optimal solution. A satisficing strategy may often be (near) optimal if the costs of the decision-making process itself, such as the cost of obtaining complete information, are considered. Interestingly, the term was not initially promoted as an economic idea, but

comes instead from administrative theory and management science, although some of the main concepts, e.g. the "economic man" were later refined as the concept gained popularity.

The concepts of Social Choice and Social Welfare from the Economics literature, as introduced by Arrow and Simon, are fundamental to the modern day idea of Computational Social Choice, surveyed by Chevaleyre et al. [CELM07]. This paper focuses on the cross over between computer science techniques which include complexity analysis, algorithm design and social choice mechanisms which include voting procedures and fair division algorithms for combinatorial domains. Of most relevance is their summary of how social choice is important in distributed resource allocation and negotiation. They distinguish two types of criteria when assessing an allocation of resources (i) criteria pertaining to the efficiency of an allocation and (ii) those relating to fairness considerations. These criteria make up a social welfare function which can be used to determine a choice when traditional economic criteria do not provide a single solution. The authors give some examples of efficiency and fairness criteria:

Pareto efficiency: An allocation Pareto dominates another allocation, if no agents are worse off and some are better off in the former. A Pareto efficient allocation is an allocation that is not Pareto dominated by any other allocation. This is the weakest possible efficiency requirement.

Utilitarianism: The utilitarian social welfare of an allocation is the sum of the individual utilities experienced by the members of society. Asking for maximal utilitarian social welfare gives very strong efficiency.

Egalitarianism: The Egalitarianism social welfare of an allocation is given by the individual utility of the poorest agent in the system. Maximising this value is a basic requirement for fairness.

Envy-freeness: An agent is envious when it would rather get the bundle of resources allocated to one of the other agents, an allocation being envy-free when no agent in the set is envious.

As we know, in distributed resource allocation, allocations emerge as a consequence of individual agents trading through a sequence of deals to exchange goods and form allocations. The question is whether this set of actions can converge to give socially

optimal outcomes and this can only be determined by evaluating the fairness and efficiency properties of every outcome.

There is further interesting work in [GO96] where Greiner and Orponen look at a set of satisficing strategies which aim to produce the optimal (minimum expected cost) method of running a set of experiments. This paper is helpful because the authors have developed previous work on satisficing strategies into a generalised approach such that if a problem can be represented as an and-or search tree, then their algorithm "pao" can produce an approximately optimal solution even when not all of the preferences / weights in the tree are known.

Finally, Economics and Computer Science begin to merge again with Robert Axelrod's book "The Complexity of cooperation" [Axe97]. Axelrod looks beyond simple cases, such as the Prisoner's Dilemma to study a wide range of issues including how to cope with errors in perception or implementation, how norms emerge, and how new political actors and regions of shared culture can develop. Whilst this is a book aimed primarily at social scientists, he focuses on using Agent based modelling to unearth some of the emergent properties of the social systems described.

2.3 Conclusions from the literature

From the initial days of mainframe computing and the introduction of Social Welfare in the late 1930s, research into resource allocation, management and fairness for both centralised and decentralised systems has been undertaken in a number of fields throughout computer science and economics. In reviewing the literature we identified a two major themes:

Firstly, there is a large body of work which describes both the design and attempted implementations of market simulations, usually with a specific focus. For example, the work by Dang [DJ03] focused on demonstrating a new clearing algorithm for multi-unit auctions that consistently performs within a bound of the optimal solution, whilst a paper by Biswas [BN05] looks at worst case performance in Combinatorial Auctions. Frustratingly, there is little concrete discussion as to how these simulations were assembled, or what their actual performance was and therefore it is subsequently impossible to build comparisons or benchmarks of either their system or ours. Similarly,

many of the scheduler implementations also focus on only a single aspect of the mechanism in hand. For example, ARMS [CJS+02b] focuses on service advertisement and discovery, whilst Nimrod is concerned with management of spare resource on idle workstations, again this makes comparison difficult.

Secondly, Authors appear to fall into one of two camps, those for centralised markets and those for decentralised. There does not appear to be a body of work devoted to comparing and contrasting the two viewpoints, such that each party is either content for the other to exist, or, of the opinion that theirs is the only solution to the problem at hand. Whilst it is true that for some types of problems a specific mechanism is best (e.g. when governments auction radio spectrum a combinatorial auction is the only sensible mechanism for doing so) there are others (financial markets, computer resource scheduling, etc.) where both approaches have strong merits. We believe that the two ideas (distributed resource allocation and CAs) can be further developed together, because in a distributed system with few trust relationships between parties, a single centralised auction does not present a good solution to solving complex distributed resource allocation problems.

The literature reviewed covers a wide spectrum of knowledge. We firstlymarketmodels considered the detail of both approaches, including papers on Centralised approaches by authors such as Sandholm and on distributed markets in Section 2.1.2 and implementations of both distributed markets and centralised ones. There are a number of key themes for distributed markets, mobility, performance and the selection of an agent platform to provide the development environment and we looked at relevant previous work for each, which informs Chapter 4, where we detail our development work.

One of the aims of this thesis is to reintroduce some of the relevant economic theory into market design. We need to understand the theortical decisions that need to be made in order to build a succesful distributed market, but also to enable a rational basis for understanding when centralised and de-centralised systems should be used, which is where our tour of economics demonstrates its importance. Papers covering Social Welfare, Arrow's Impossibility Theorem and Satisficing, three important concepts in building our distributed system and the comparison thereof have been reviewed in Section 2.1.3. We also identified some "must have" efficiency and fareness criteria—because by using the same metrics that economists use we can be sure we have developed an effective market. These are (i) Pareto efficiency, (ii) Utilitarianism, (iii) Egalitarianism and (iv) Envy-freeness, discussed in Section 2.2 on page 50.

Having identified the economic concepts and the work that introduced them, in Chapter 3 we begin with a discussion of Models and Algorithms to to apply our knowledge to the problem of resource allocation.

Chapter 3

Discussion of Models and Algorithms

3.1 Introduction to Market Models and Approaches

In this chapter we will explore the economic models we have used to structure our approach for solving the bundling problem. This allowed us to develop a prototype distributed auction with the same economic foundations as a centralised Combinatorial Auction, ultimately implemented as our Multiple Distributed Auction (MDA) system, which is explained fully in Chapter 4.

Comparing centralised and decentralised systems could be perceived as an apples and oranges comparison. However, both types of market are solving the same problem—that of how to produce a resource allocation according to specific criteria. We have built a platform to support the comparison of these two techniques which uses a common test data set from the CATS suite and derives a base line from the widely available CASS solver. We consider why these two tools provide a sound basis for the generation of data for comparing the two approaches.

We begin with a discussion of the economic properties necessary for a sound market and review the Vickrey-Clarkes-Groves mechanism as a template for our market designs.

3.2 Auctions and clearing techniques

3.2.1 Key Economic Principles

There are a number of key economic principles which come together in market design to ensure that all participants receive the optimal outcome from the market. We briefly describe them as follows:

Dominant Strategy

A strategy for buying or selling goods is considered dominant if, regardless of what any other players do, the strategy earns a player a larger payoff than any other. It will therefore always be better than any other strategy for any profile of other players' actions.

The concept of a Dominant Strategy is important in market design because it ensures the participant is going to maximise their outcome from the mechanism. Therefore, in developing our markets, we need to ensure we encourage use of the participant's Dominant Strategy.

Nash Equilibrium

The concept of Nash Equilibrium was proposed by John Nash in 1950 in his work "Equilibrium Points in n-Person Games" [Nas50].

> "The concept is an n-person game in which each player has a finite set of pure strategies in which a definite set of payments to the n players corresponds to each n-tuple of pure strategies, one strategy being taken for each player."

i.e: Alice and Bob are in Nash Equilibrium if Alice is making the best decision she can, taking into account Bob's decision, and Bob is making the best decision he can, taking

into account Alice's decision. However, to make the best possible decision each party will need to understand the others preferences and strategy for their decision making process; this is known as "full revelation".

A Nash Equilibrium is important because it means that the expected average payoff for the participants in a mechanism is at least as large as that which would be obtainable by any other strategy.

Satisficing

Satisficing is a decision-making strategy which derives an adequate, rather than an optimal solution. A satisficing strategy may often, in fact, be (near) optimal if the costs of the decision-making process itself, such as the cost of obtaining complete information, are considered in the overall cost calculations.

Herbert Simon combined the terms "satisfy" and "suffice" to promote the term "satisficing" [Sim55, Bro04]. He drew inspiration from humans, as we are usually able to maximise the outcome of a decision: (i) we rarely know all the relevant probabilities for different options/outcomes available to us, (ii) we are often unable to make decisions to a sufficient level of precision and (iii) humans become old and their memories unreliable, corrupting our basis for decision making!

However, despite only being partially rational in their decision making, most humans are relatively successful, and Simon concluded this was because we treat goals as something not to be maximised, but as a constraint. We therefore always try to meet the minimum level across a specific set of goals, but thereafter, we may arbitrarily apply our focus to other goals, not necessarily that which will maximise our benefit from the decision.

> "The most common application of the concept in economics is in the behavioural theory of the firm, which, unlike traditional accounts, postulates that producers treat profit not as a goal to be maximised, but as a constraint. Under these theories, a critical level of profit must be achieved by firms; thereafter, priority is attached to the attainment of other goals." [Sim55]

Wikipedia states that "Simon, once explained satisficing to his students by describing a mouse searching for cheese in a maze. The mouse might begin searching for a piece of Gouda, but unable to find any would eventually be satisfied and could suffice with any piece of cheese, such as cheddar." (From `http://en.wikipedia.org/wiki/Satisficing` 5-Apr-2009)

Incentive Compatiblity

When placing their buy or sell offers, we need to encourage participants to tell the truth, so that the market maker, who makes the allocation decisions in the market, has the best chance of making an economically sound decision. This property is defined as being "incentive compatible". Assuming there is no price collusion or false name bidding in the market, a second price auction is an example of mechanism that is incentive compatible (Chapter 1 of "Combinatorial Auctions" [CSS05] provides an excellent overview of auction types).

> "In mechanism design, a process is said to be incentive compatible if all of the participants fare best when they truthfully reveal any private information asked for by the mechanism."

It is worth noting that there are different degrees of incentive compatibility. For example, in some games truth-telling can be considered a dominant strategy (as long as everyone else will tell the truth).

Pareto Optimality: Optimal allocation for whom?

As Wellman, Walsh et. al discuss in [WWWMM98] "If there is some way to make some agent(s) [economically] better off without harming others, it should be done. A solution that cannot be improved in this way is called Pareto Optimal."

> "Pareto optimality, is an important concept in economics with broad applications in game theory, engineering and the social sciences. The term

is named after Vilfredo Pareto, an Italian economist who used the concept in his studies of economic efficiency and income distribution.

Given a set of alternative allocations of, say, goods or income for a set of individuals, a movement from one allocation to another that can make at least one individual better off without making any other individual worse off is called a Pareto improvement. An allocation is Pareto efficient or Pareto optimal when no further Pareto improvements can be made. This is often called a strong Pareto optimum (SPO).

A weak Pareto optimum (WPO) satisfies a less stringent requirement, in which a new allocation is only considered to be a Pareto improvement if it is strictly preferred by all individuals (i.e., all must gain with the new allocation). The set of SPO solutions is a subset of the set of WPO solutions, because an SPO satisfies the stronger requirement that there is no allocation that is strictly preferred by one individual and weakly preferred by the rest (i.e., no individual loses out, and at least one individual gains).

A common criticism of a state of Pareto efficiency is that it does not necessarily result in a socially desirable distribution of resources, as it may lead to unjust and inefficient inequities." (From `http://en.wikipedia.org/wiki/Pareto_efficiency` 17-Dec-2008)

Let us consider the attraction of Pareto optimality. Given the choice between optimal and non-optimal allocation, the former is likely to be selected as more desirable, yet in many real world situations optimal allocation is rarely achieved and it is often deemed preferable to have some slack in the system.

Additionally, one problem with Pareto optimal solutions is that they may not be socially just. I.e. if Alice and Bob are in a market, the Pareto optimal solution may be to give all the goods to Bob, but this may be socially unfair on Alice.

Why does this happen? The problem is that for the Alice and Bob market we are unable to tell their true preferences exactly as they are expressed incompletely. This means that the auctioneer making the allocation decision for Alice and Bob has an element of his search space of possible solutions which is incomplete and may preclude him from identifying the best possible allocation.

When considering the Pareto efficiency of the Alice and Bob market, we may find that there are multiple Pareto efficient outcomes (all goods to Alice, all goods to Bob,

or share the goods) and this means that we have not found a sole Pareto superior outcome—rather, we have a Pareto frontier.

How do we resolve this issue? In order to make a choice between the different Pareto efficient outcomes the market will need a further ranking mechanism. The question arises: "optimal for whom?" because in this instance, true Pareto optimality at any social expense is not that which is required. A better solution would be one which is Pareto optimal and which maximised the social welfare of the participants, the final economic principle which we consider in Figure 3.2.1. We can visualise the problem in Figure 3.1.

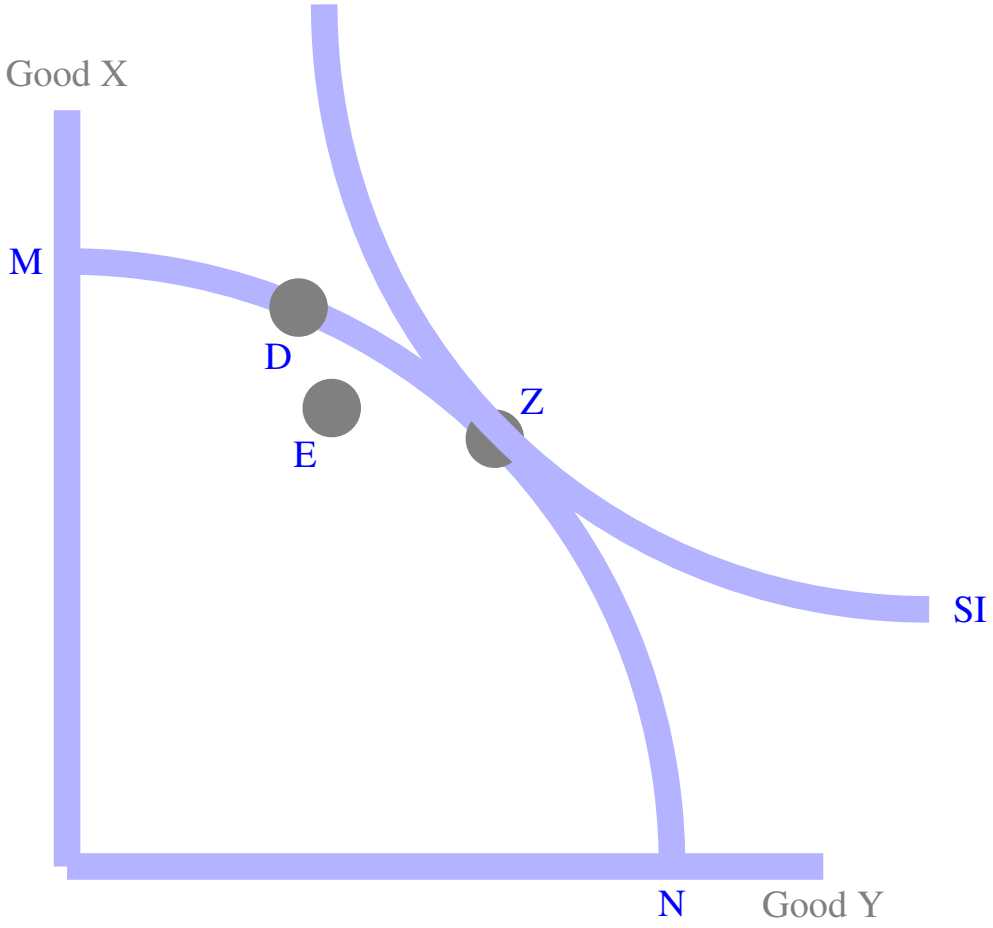

Figure 3.1: Visualisation of Pareto and Social Optimality converging

The graph shows two lines. MN is a social utility frontier, that is the range of utility values that are considered to be Pareto optimal. Point D indicates a scenario where production and consumption are efficiently matched, whilst point E lies inside the social utility frontier and indicates inefficiency. Although all the points on line MN are Pareto efficient, only point Z identifies a scenario where social welfare is maximised,

with line SI representing the social injustice that the allocation might cause and point Z lying at the intersection of a Pareto optimal solution and minimum social injustice.

We can relate this back to market design, by arguing that in the commercial auction space the aim should be to achieve the outcome with the highest overall utility for the seller, bearing in mind that on some days, a trader who makes one sale is better off than one who makes none, and one of the problems presented by Combinatorial Auctions is that they are unable to provide us with solutions to our markets without complete and truthful information about all parties preferences, as we shall see in our discussion of Vickrey Auctions in 3.2.2.

What happens if there is no interception of the pareto optimal and social welfare functions? In this case we have an allocation which is economically inefficient, similar to point E and therefore the buyer or seller is unlikely to want to participate. If there are multiple intersections to the two lines then we have defined a function that produces multiple equivilent outcomes. This would not however be considered a well formed social welfare function—in order to become well formed, the criteria used must ensure that the system is able to make a selection between possible outcomes in order to maximise the social welfare of the decision.

Social Welfare

Welfare Economics is concerned with understanding the social impact of economic decisions and can be used to ensure that these decisions do not create unnecessary (or unwanted) social injustice.

In our discussion of Pareto optimality, we concluded that we required a ranking function which would permit our resource allocation algorithm to rank the outcomes such that we would have optimal social welfare. This notion is described in the literature as a Bergsonian Social Welfare Function, invented by Abram Bergson in 1938 with the objective:

> "to state in precise form the value judgements required for the derivation of the conditions of maximum economic welfare" [Ber38]

However, we can observe that it is difficult to create Social Welfare Functions which produce fair and qualitative rankings because the comparison of utility is an inexact science. Consider the following problem: Alice and Bob both consider themselves to be "happy". Who is the most happy, or are they equally happy? We cannot tell because happiness, or indeed utility, or indeed welfare/well-being does not have cardinal properties (Is happiness an 8 or a 10?).

This thought-experiment is the basis of Arrow's Impossibility Theorem[Arr50], which provides four reasonable criteria that no consistent Social Welfare function can satisfy:

Unrestricted domain: This means that the Social Choice Rule must be able to incorporate any pattern of individual preferences.

Pareto principle: This means that the chosen outcome must be Pareto efficient.

Independence of Irrelevant Alternatives: This means that no set of rankings is affected by a change in another set of rankings.

No dictator: This means that no single individual can decide the outcome.

Therefore, given incomplete information and a single point of decision making, it is possible to determine a number of Pareto-efficient outcomes, but not a single Pareto-optimal outcome. To determine that, we would have to have a Social Welfare function to evaluate the various options, which is not possible in a single decision maker based mechanism, as established by Arrow's Impossibility theorem.

The ability to produce an allocation whilst adhering to the criteria given above is one of the key benefits of using distributed markets, and it is important for us to revisit these properties as we progress through the thesis.

3.2.2 Vickrey-Clarkes-Groves Mechanism

A Combinatorial Auction is essentially a weighted set packing problem [BN05] which is known to be NP-hard. There are various ways of characterising and solving this problem which we reviewed in detail in Section 2.1.2, but a key mechanism often referred to as an architectural starting point is the Vickrey Auction.

The book “Combinatorial Auctions” has an intriguing chapter title: “The Lovely but Lonely Vickrey Auction”[CSS05, Ch.1]. The Vickrey Auction is a sealed-bid auction where bidders submit their bids without any knowledge of how the other participants are bidding. It was invented by William Vickrey in 1961.

The Vickrey Auction focused on auctions which sold a single, indivisible good. It is a second-price sealed-bid auction, where the winner pays the amount of the second highest winning bid. It is said to be “incentive compatible” in Mechanism Design if all of the participants fare best when they truthfully reveal their private information—typically the value of the good to them to bid truthfully. However, if the bidder has a demand for more than one of the good, then this approach will cause them not to bid their true valuations for the goods.

A modification to the Vickrey Auction to support multiple goods and to maintain the incentive compatibility of the market was therefore needed. Two further mechanisms by Clarke and Groves can be combined to provide these features and make a system known as the Vickrey-Clarke-Groves (VCG) mechanism in which the winner pays the second highest price bid.

The key elements of the VCG mechanism are:

Dominant Strategy: In a VCG, reporting their true valuation is a dominant strategy for a bidder. This means that they will get the best (most profitable) outcome by following that strategy. It also means that the cost of running the auction is less (for the bidders) because they no longer have to invest energy into determining the strategies of other bidders and working out how to compete with them.

Scope of Application: It is possible in a VCG to set filters and rules without affecting the economic properties.

Average Revenues: The VCG mechanism guarantees that the average revenues from the system shall not be less than any other efficient mechanism. This means that it has been designed so as to ensure there are no drawbacks to using it for the buyers and/or sellers.

The key downsides to the mechanism are:

Low Seller Revenues: Non-Monotonicity: "better" bids don't entail higher revenues because the seller's revenues are non-monotonic with regard to the sets of bidders and offers. Consider bidders A, B, and C, and two goods, Y and Z. A bids £2 for the package of Y and Z. B and C both bid £2 each for a single item (bid £2 for Y or Z), as they really want one item but don't care if they have the second. Now, Y and Z are allocated to B and C, but the price is £0, as can be found by removing either B or C. If C bid £0 instead of £2, then the seller would make £2 instead of £0. Because the seller's revenue can also go up when bids are increased, the seller's revenues are non-monotonic with respect to bids.

Collusion: Losing bidders can collude to force pricing down and reduce profit.

False-name bidding: A bidder can bid multiple times, using different aliases, in order to get better pricing and increase their chance of success.

The VCG is important for solving resource allocation problems because it is the major market mechanism in use which gives the relevant economic and game-theoretic properties for a market that is Pareto-optimal and maximises social welfare. However, it has some serious limitations which impact its usefulness in the purest form, as we now explain:

Ausubel and Milgrom [CSS05, Ch.1] characterise the VCG as a "lovely and elegant reference point—but not as a likely real-world auction design. Better, more practical procedures are needed." They propose further designs to tackle the deficiencies stated above, including an iterative auction procedure combined with an ascending proxy bidding procedure known as the "Clock-Proxy Auction" [CSS05, Ch.5]. This process is very similar to the one currently adopted by the EBay auction house.

In order for our comparison to be valid, we need our MDA mechanisms to exhibit identical economic properties and we review our compliance in Chapter 4.

With regards the CASS solution, it is provably optimal: It finds a solution that maximises the social welfare as do the widely known BidTree and CPLEX solvers. However, there are no published verifications of the algorithm but we judge its accuracy from the wide publication of papers which reference it and from private communication with the author, Leyton-Brown, on this subject:

> It's possible that there will exist solutions that tie (i.e., that yield the same

> SW with different bids); in this case, the different methods could find different sets of bids. I've done experiments confirming that CASS really does find the optimum, checking against CPLEX (I've never been given code for BidTree or CABOB). So have others, e.g., Craig Boutilier. As far as I know these results haven't been published. (Source: Email received August 6, 2008)

3.3 Distributed Market Models

There are many algorithms for solving combinatorial auction type problems in the literature, where it is generally accepted that the optimal solution to a resource allocation problem may be found using a combinatorial auction and although the theoretical cost is NP-hard, even relatively naïve solvers like CASS [LB03] can handle many problems quite rapidly, while the most sophisticated services like those available from CombineNet [San07] can process most problems quickly thanks to a combination of a range of clever heuristic techniques and specialised bidding languages that help to reveal more about bidder preferences.

However, this line of research poses a new problem, that of comparison, and it is this that Kevin Leyton-Brown set out to solve when he created the Combinatorial Auction Test Suite (CATS) in 2000.

Given that in this work we are building a comparison of a centralised auction system and a decentralised market system, if one conceptualises a discrete space of algorithm complexity then there are a number of steps available as we shift in the market on an axis of complete information held in the auctioneer to minimal information and a second axis from one auctioneer to many, comprising of: (i) Entirely centralised combinatorial auction, then (ii) An agent based distributed market, with synchronous rounds, and finally (iii) An agent based market with asynchronous trades.

We believe it is necessary to consider these axes of resource allocation techniques because there are three reasons to dissent from the view that combinatorial auctions are the only solution required for resource allocation and that it is therefore a solved problem:

1. In many practical situations, the resources and the bidders are distributed, so that the centralised mechanism that is the necessary property of a combinatorial auction is inappropriate [PS04].

2. Runtime: An anytime (sub-optimal) algorithm with predictable runtime may be preferable to an optimal algorithm with an unpredictable runtime. One approach to this is [DJ03], which describes an anytime polynomial algorithm that guarantees to be within a bound of the optimal, such that each step of the algorithm reduces the bound.

3. Finally, for the sake of resilience (another aspect of timely delivery) the potential single-point-of-failure that is intrinsic in CAs may also be undesirable.

In order to complete a valid comparison, our experimental approach has been to use a common dataset, produced by the CATS algorithm, as the input into all of the simulations. We have also sought to maintain a comparable execution platform, with comparability of processor type and execution environment.

Throughout the period of this work, we have considered a number of different frameworks, test platforms and simulation systems and document them in detail in the following pages. They are:

1. Brickworld - our initial simulation

2. MDA - Multiple Distributed Auctions, implemented in Repast

3. MDA AgentScape - The MDA system, implemented in AgentScape

3.4 Brickworld - Initial design concepts

BrickWorld was an exploratory prototype model of a distributed combinatorial purchasing system which utilises multiple distributed single item auctions (MDAs). Whilst the particular method of implementation was ultimately unsatisfactory, it provided helpful insights into the implementation of markets and was a key part of the evolution of our designs.

Inspired by Lego Bricks which usually have a high level of (multi) dimensionality, in terms of colour, shape, dimension and applicability to different situations we coined the term "BrickWorld" to represent a market that traded "Bricks" which act as tokens representing the many different and complex selection criteria that both a Combinatorial Auction (CA) and a distributed bundling system should be able to handle.

Where real world traders are able to purchase goods for resale from a wide variety of markets they are often able to increase margins by providing value-added services or combinations of goods otherwise previously unthought-of, or which may be tricky for an individual to acquire. The TraderAgents in BrickWorld took on a similar role in that they assemble bundles to order and speculatively.

We investigated a number of issues:

- Where do traders get their budget from? Can they go into deficit?
- Can traders take positions on their goods?
- How do traders make decisions about what to buy or sell?
- How to manage message passing in a distributed environment—would the overhead kill the efficiency gains of the system?
- Longevity of goods in the market place and how to handle perishable goods.

3.4.1 Brickworld Development

We developed a Java Remote Method Invocation (RMI) based AuctionServer to which AuctionClients connect. An AuctionClient can either be a BrickFactory (which generates Bricks available for sale) or a Participant, who will attempt to purchase bricks.

The design and implementation was such that any number of auctions could be held concurrently and a single "Auction" would handle the sale of a single Brick. Thus a Participant in the market would have to participate in a reasonably large number of auctions in order to assemble a collection of different bricks. The system facilitated the synchronous updates of all the traders who were monitoring an auction.

Traditional search systems attempt to provide solutions through the use of heuristics to prune their search trees. One mechanism for pruning is to filter elements based on their attributes. A key improvement of BrickWorld over other trading systems is that it was designed such that the multi-dimensionality, (i.e. number of attributes) of the objects being traded is not a significant factor in the time taken for the optimum allocation to be made. This benefit arises because Brickworld is a distributed market and solutions are not determined using a centralised search algorithm. Bricks had many attributes, initially having Colour, Weight, Height, Length and Width.

We focused on two types of auction, the Sealed Bid and the Open Cry auction. BrickWorld was structured so that an Auction has to implement a number of key elements of functionality (as defined in the BrickAuction interface). This means that any number of different types of auction mechanisms can be implemented and simply "slotted in" to the BrickWorld system as required.

The Sealed Bid auction is one which implements a first price sealed bid auction, such that the item is put up for sale, all bids are received before a fixed (pre-announced) point in time and then the highest bid is determined and announced as the winner. The Sealed Bid auction was used as a simple harness to test the code and so the next development was the more complex OpenCryAuction.

The OpenCryAuction class is designed to run continually. Clients connect to the auction at any time and begin to observe the auction item. The auction runs for a set amount of time and every observer is notified if the auction item changes (i.e. typically if the value changes). At the end of the auction the winner is computed (i.e. the highest bidder) and the auction closes.

3.4.2 Auction closing and Settlement

We were keen map to the experience of traditional human led markets and auctions to our simulation as much as possible and a big issue was that of when to close the auction.

In a real world auction the auctioneer is able to observe the auction room and all the potential bidders nearly simultaneously. Even if the auctioneer has to handle telephone bids, a human has sufficient sensory input that they can (with a small margin of error)

determine if there will be any "last minute" bids either in person or via other means and thus execute the classic "going going gone" auction ending.

In electronic computer driven auction mechanisms that discretion is not available and there needs to be a firm fixed end point. However, this leads to a syndrome of gazumping in which many bids are submitted in the final and closing stages of the auction, with bidders attempting to ensure that they do not expose any of their internal valuations to the competition. Ebay exhibits excellent examples of this behaviour.

From initial implementations of fixed markets with defined closings, we chose to refine the implementation of BrickWorld and our subsequent MDA implementation so they operated as series of multiple continuous double auctions, each dealing with a single type of good. Rather than closing at a specific time the markets continue to make the good available until sold, with buy and sell shouts being made until pricing converges and a successful trade is completed.

Our simulations are implemented using electronic tokens that represent actual goods. As such it is envisaged that in a real implementation, an authorisation message would be passed back to the Grid Scheduler which can then complete the service delivery. A separate system is required for the passing of goods, or authorising access to services following from the auction.

This focus on "tokens" also helps deal with market liquidity issues, as in an MDA system a trader can become insolvent and there is potential for value to become "locked". In the absence of a Bona Vacantia [1] system for the gathering of assets of the deceased, suppliers with sold, but unredeemed vouchers may decide to resell them after an appropriate period.

We also briefly evaluated the issues surrounding the security and robustness of the market and how these issues might be mitigated. For example in a computer system one bidder might launch a denial of service attack on another to stop their bids being received, or indeed, the auctioneering software may be hijacked and replaced with a "corrupt" auctioneer whom favoured one particular bidder. Whilst these things can happen in real life, a computer network makes it much easier for them to occur and much harder to observe. We look at various related works in Section 2.1.4, page 41, of

[1] Bona Vacantia literally means vacant goods and is the legal name for owner-less property in the UK which passes to the Crown.

our literature review and feel that in order to ensure integrity in the market mechanism it would be important to ensure such strategies are implemented in any production system.

3.4.3 Evaluation Strategies

A number of derivative works from this thesis have been published and we are continually challenged on the question of comparing the effectiveness of two very different approaches.

On the one hand CA is a single auction with complete knowledge that, given enough time will compute the optimal allocation given the bidders' preferences and a small enough problem. On the other the MDA is a system of distributed auctions that are connected indirectly by the simultaneous participation of bidders in more than one auction and that will result in the creation of many bundles that the trading agents view as saleable. CA appears to offer mathematical certainties while MDA offers possible solutions, although MDA has an undeniable advantage in one aspect in that it appears to be scalable. In figure 1.4 there is a point at which it is no longer feasible to compute the outcome of a bundling problem using a CA, but where it would be possible still to determine an outcome through the use of a distributed trading system. Below that point, the use of perfect information would mean that a CA could determine the Pareto Optimal solution and therefore the trading system should be expected to achieve a similar result. Beyond that point, an allocation can be achieved, but may not be Pareto Optimal.

An analytical approach to comparison seems infeasible, at least, given current understandings of the problem so empirical techniques must be adopted to evaluate the performance of the system in terms of the optimality and speed with which it bundles but also the allocations achieved.

By using identical request feeds to a CA and MDA (through the use of CATS), from which they will produce bundles, a cluster analysis to observe the proximity of the solutions may be achieved. The results should be useful even with ZIP-style agents [CB98] because the structure of the resource allocation process (i.e. centralised or decentralised) will have significant impact on the bundles assembled.

The first step in evaluating the two techniques was to compute a base line performance model of the currently available solvers, such as LPSolve [2], CASS [LB03] and those implemented by Dang [DJ03]. This empirical analysis produced some interesting results, in that, whilst the performance model for a CA is anticipated to be NP-Hard, we actually find that through improved use of search heuristics, many of the computations are able to complete in linear time, whilst a few problems take a very long time to compute.

Secondly, we developed the need to understand the performance model of a distributed agent scheduling system, which resulted in the implementation of our MDA system.

In moving toward the use of empirical evaluation techniques our results will become subject to the vagaries of the computational environment that is being used. To this end the agent based systems are being measured using not only auction rounds and time, but also clock ticks, as suggested in work on comparisons of distributed constraint optimisation problems (DCOPs) by Mailler [Mai05].

In evaluating MDAs, the following metrics were adopted:

- Firstly, the performance model of the agents bidding for items is considered. From previous work by TAC participants and others [DF03, HJ02, AHDJ01, KKD$^+$04, TDTY04] shows that the performance of the auctions can be improved by heuristics and domain specific knowledge. However MDAs operate in multi-domain environments and it is important to understand the performance of a suitably non-domain specific agent and measure its ability to join and monitor auctions, manage spending and assemble bundles of goods for which there is a market within a specific time frame, so that we can understand generic trading performance.
- Secondly, monitoring the auction and following a number of important metrics, such as (i) number of nodes, (ii) auctions, (iii) bids, (iv) bidders, (v) time for auctions to complete (vi) and length of queues for items awaiting auction.
- Thirdly, determining the quality of the result obtained by the trading system allows us to determine the level of effectiveness of the process which is particularly useful when the complexity of the bundling problem lies above the level of results achievable with a CA.

[2] LPSolve is available at `http://lpsolve.sourceforge.net/` 24-04-2009

Finally there is a need for a CA to complete in "reasonable time" because one of our objectives is to produce resource allocations which are not just good quality, but are also relevant to their requestors. To define reasonable, the bundle request data has a range of simulated deadlines against which the bundling system's speed can be tested.

3.4.4 Conclusions from Brickworld

The Brickworld implementation fell short because it was not feasible, with a single developer, to develop a complete and robust auction system from the ground up in the time available and we felt that it would be more advantageous to utilise existing similation frameworks, such as Repast and AgentScape so that our development time could focus on the auction logic.

However we were able to identify a number of elements in our development which allowed us to build a more successful MDA system. These included (i) the market closure, settlement and handover process, (ii) integrity of the market and (iii) an evaluation strategy.We also identified a number of areas for further investigation, including:

- Whilst BrickWorld would work effectively on the basis of "just-in-time" ordering it is also intended to investigate pre-emptive purchasing strategies. It is clearly desirable for a TraderAgent to be able to deliver resources required with minimal delay, whilst undesirable for the agent to purchase too many advance resources without customer commitment for their purchase. In the MDA system, we adopted the ZIP trading strategy to manage this process, implemented in the JASA libraries [PMPM06].

- If a TraderAgent ends up with surplus capacity then it would make sense for the trader to be able to sell their surpluses to other traders in the same set of auctions, a function we implemented in the MDA architecture.

- There is no reason to restrict resale as there is no motive to hoard resources, nor to sell them at a loss. Feedback and re-entry mechanisms for an MDA system need to be considered in future work.

- We could increase the robustness and security frameworks in the BrickWorld system, for example, to include protocols and architectures derived from work by

Franklin and Reiter [FR95], Stubblebine and Syverson [SS99], and we propose that this be considered in future work.

Finally, we have identified that there needs to be consideration of an optimisation technique which would be borrowed from functional programming: memoization [Mic68]. Consequently, instead of computing the optimal allocation each time it would be possible, over time, to look it up. It might even be acceptable to return a previous allocation if the preferences were close enough to an earlier case (subject to some proximity bound on the distance from optimality and to analytical continuity). At this point it becomes apparent that the CA has acquired some market memory too and that within acceptable deviation from the optimal solution the results of the two approaches could be largely indistinguishable — except that the MDA approach will continue to function as the complexity of the bundling problem increases, whilst a CA will not.

Chapter 4

The MDA, or Multiple Distributed Auction system

We have developed a resource allocation system that uses Multiple Distributed Auctions (MDA). We will outline the simulation system and test models and describe the economic and algorithmic properties of the systems and software used. For MDAs we consider both the centralised (MDA) and decentralised (MDA-AgentScape (MDA-AS)) implementations in the context of these economic properties and demonstrate how they provide a sound basis for comparison.

4.1 MDA Trading System Architecture

It is well known that clearing a CA over bundles of heterogeneous items is an NP-Hard task. In contrast, trader agents in an MDA operate in multiple single-item auctions, to achieve resource allocation, even under circumstances of incomplete preference information.

The MDA vision, as shown earlier in Figure 1.4, is that a user (ClientAgent) composes their request for computing resources, specifying a number of different components that will be needed (for example, a data set, a software package and an amount/type of CPU). The request will probably also contain other restrictions, such as cost, quality of service, time to delivery and the level of completeness required, for example whether a

partial answer is acceptable. The request is then handled by the following components, which communicate with each other through Remote Method Invocation (RMI) and Web Service interfaces:

- The ClientAgent issues a call for tenders, using an order-board style mechanism, which is met by a number of TraderAgents. As well as considering the quotation responses on the basis of price, ClientAgents may also consider the ability of the TraderAgent to deliver the necessary requirements (E.g. on the basis of a reputation model managed by the Auction house.). The ResourceAuction contains a reputation model for the resource providers, which could then be used by the TraderAgents to present an indication of their ability to deliver.

- In order to meet the requirements of the ClientAgents, TraderAgents purchase items from a collection of continuous double auctions (CDAs) that take place on several MDA system nodes. Nodes are interconnected and Traders are free to monitor and enter auction processes, placing bids as required to meet their clients requirements.

- The TraderAgents are profit-motivated and their objective is simply to acquire resources, combine them and sell them to ClientAgents. TraderAgents will be limited in that they may not trade insolvently overall (if they do become insolvent, they have failed and will die), but they will inevitably acquire resources that they cannot then sell.

- Traders can adopt different fulfilment models, either speculating on client demands and purchasing in advance or waiting for a requirement to arise, and then attempting to create a suitable bundle to match.

- When the MDA environment is running, the Seller will take resources for sale and feed them to the auctions. They can then monitor the resources absorbed by the market and report those back to such as operate in multiple single-item auctions to achieve effective resource allocation without any of the optimality guarantees of a CA.

The MDA system should be completely distributed and traders discover, monitor and purchase items from a large number of different auctions. Each node in the system therefore supports the necessary functionality to support self-registration by other

nodes, so that a web of nodes can be deployed and then navigated via Traders and Clients.

We set out to create a distributed market mechanism for solving bundling problems. That is, given a requirement for a number of Buyers Bn, to purchase multiple tuples of goods from the overlapping set X,Y,Z,P,Q can we successfully provide a mechanism that allows the buyers to make their purchases within a Socially Optimal manner?

Recall also that a key objective for us is to be able to compare MDA critically with the centralised system. Consequently, the MDA is derived from the same problem specification and operates on the same datasets.

Finally, we sought to build a resource allocation system which could handle scenarios with (i) incomplete data and (ii) communication complexity in a more robust manner. This specifically meant that we required a solution which was completely decentralised with no single points of control or decision making, hence a system of distributed auctions.

4.2 Overview of MDA

For simulation purposes our implementation builds a centralised source of knowledge, the "Oracle", that processes the bid request files produced by CATS [LBPS00]. We are utilising two CATS datasets (L2, Scheduling) in our work and each distribution is used to produce a set of available bids and items which are distributed to the traders. From a random starting position, the agents (implemented using JASA) then buy and sell the goods, through many rounds, until the market stabilises and the level of trade tends to zero, indicating that none of the available items matches the bids being requested.

In our development, we observed that the market environment initially experienced a high level of activity as buyers had complete budgets and full procurement requirements whilst similarly, sellers had complete stocks and therefore there was a high degree of requirements matching. As time passed the ratio of trades per time increment was reduced, the market stabilised and we observed that after all of the goods had been traded approximately twice there was minimal new trade. Therefore, after ensuring that the market has stabilised by ensuring all goods have passed through the

market once, the simulation then continues for as many ticks again as have already passed before we close the market.

We developed the MDA as a distributed type of multiple item auction, whereby the system runs one continuous double auction for every class of good (thus eliminating the need to bid for combinations of goods in a single auction and greatly simplifying the market for both traders and auction managers). An MDA has the following core components, shown diagrammatically in Figure 4.1:

MDA Manager: The MDA manages the auctions and is responsible for telling traders where (i.e. which CDA) they can trade their resources. It holds pointers to all CDAs and if a CDA is requested that is not available then it will be created by the MDA.

CDAs: A continuous double auction where a resource is traded. Only one type of resource can be traded by a CDA and for every CDA there will be a corresponding resource type. CDAs are created by the MDA Manager.

Goods: These are the units that are traded. They are simply an identifiable reference to a resource, and the corresponding price information.

Oracle: Responsible for handing out bundles to traders on demand. As such, varying the Oracle's output defines the supply (and demand) in the market. Bundles can be both requested from the Oracle and returned to it (if they cannot be purchased/sold). All transactions are reported to the Oracle, so it maintains information and history about the market participants.

Trader: Responsible for retrieving bundles from the Oracle and trading them. Traders who fail to trade their bundle within a given number of rounds must return them to the Oracle and request a new bundle — ensuring the market does not contain too many extra-marginal traders.[1] At any one time, Traders can be buyers or sellers depending on the type of the bundle received from the Oracle. Traders can switch state (from buying to selling) and each time a trader receives the next bundle to process from the Oracle it determines which role it should play, based on the quantities specified in the bundle (positive quantities indicates a need to sell, negative a need to buy). A Trader does not intentionally speculate on goods and they will not purchase more goods than they need to complete a bundle.

[1] Buyers who have paid less than the equilibrium price and sellers who are selling for more than the equilibrium price.

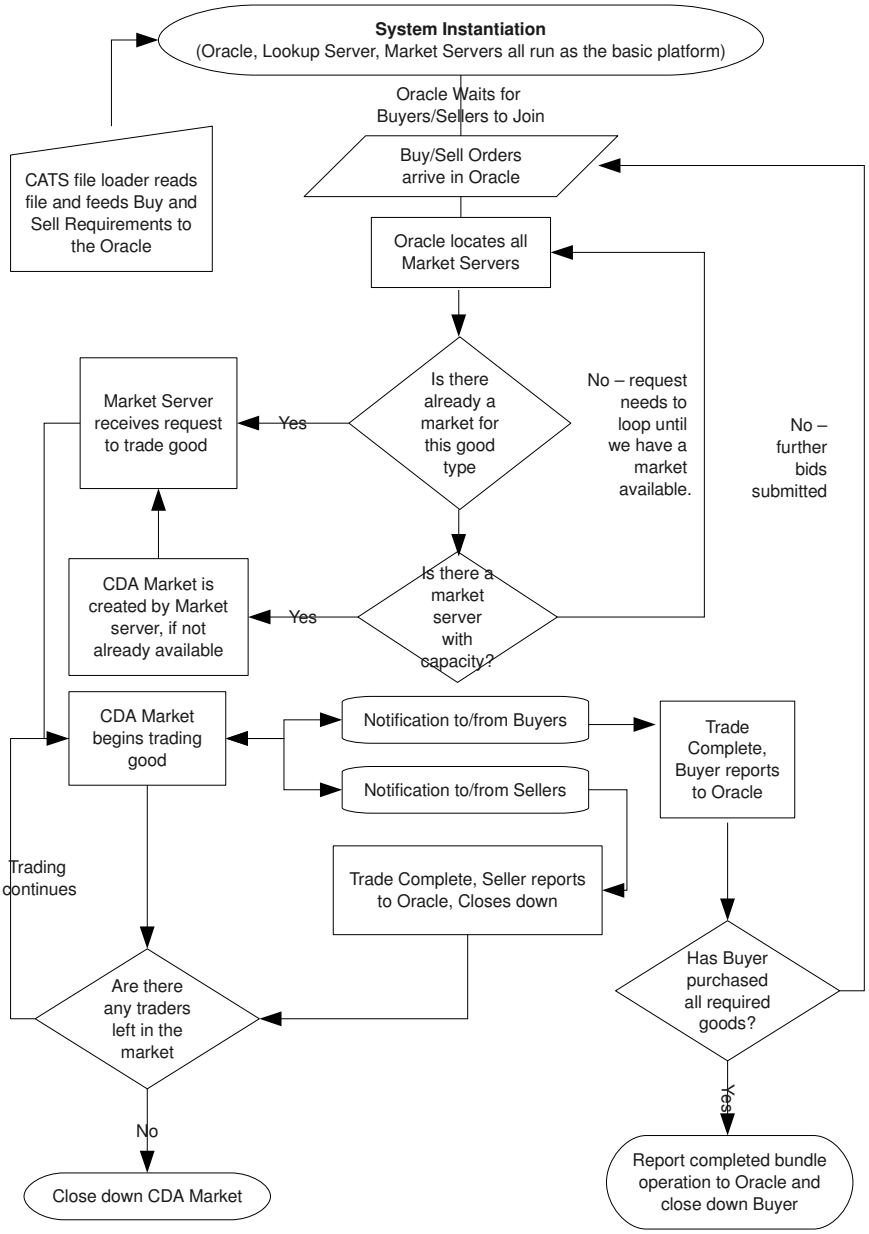

Figure 4.1: MDA Instantiation Flow

SubTraders: A Trader is responsible for assembling a bundle of goods. To do this, it participates in many CDAs, through a mechanism of SubTraders. These are agents that have the task of buying or selling a single good within a specific price range.

The combination of one Oracle, one MDA and an array of traders and CDAs makes up an MDA market. These components come together to create a Repast Model, which enables the collection and management of statistics and the control of the simulation. The input into the model is the common dataset generated by the Combinatorial Auction Test Suite (CATS). CATS generates a set of goods, and a price, for each bundle. A sample data file is reproduced in Figure 4.2:

```
goods 5
bids 15
dummy 0

0         1211656  0        3        #
1         851593   2        #
2         3927158  0        1        2        3        #
3         665147   3        #
4         2957271  0        1        3        #
5         2077835  1        2        3        #
6         4026591  0        1        2        3        4        #
7         660628   0        1        #
8         1580815  0        3        4        #
9         2052298  0        1        2        #
10        996370   3        4        #
11        2806870  0        1        4        #
12        2810633  1        2        3        4        #
13        2752827  0        2        3        #
14        334974   0        #
```

Figure 4.2: Sample CATS Data File

Each row show in Figure 4.2 describes a bundle. The first field is an integer serial number, the second the value bid for it, and the remaining fields are then the actual elements of the bid. For example, line 6 gives a bid value of 4026591 for the bundle of elements 0,1,2,3,4.

Each good can only be sold once, and therefore we can simply determine that the maximum value achievable from this set of bundles would be bids 1, 3 and 11, for total revenue 4323610 selling all of the available goods (0,1,2,3,4).

From this, you can see that for each bid, only each bidder's public valuation is expressed. CATS does not give us any indication of the bidder's private valuation, or indeed, any reference to public or private valuation from the seller of the goods. This is understandable, because for a combinatorial auction all the auctioneer must do is maximise the monies earned from selling the bundles, which it does by choosing the maximum public valuations from the buyers

However, for a continuous market, such as the MDA, there will be many rounds of bidding and the opportunity for both buyers and sellers to vary the price. Therefore, we must construct the public and private valuations required for buyers and sellers, and our approach for doing this has been as follows:

- Open the file, and loop through the lines until we find the first line that has a bundle description.
- Loop through the line. Get the bid value and all the elements in the bundle.
- For the buy bundles, the budget price for the goods is set to be the total buy value for the bundle, divided by the number of goods (i.e. it is shared equally.)
- Then create the sales bundles. Get all the resources from the resourcesRequired.keySet(). Create a new bundle, and give it one single item. Set the private valuation of the sell bundle to be the average maximum budget available (i.e: set it arbitrarily high, so it can be negotiated down, but not too high that the negotiation process takes for ever).

Once the bundles have been read from the file and the different valuations have been computed, they are boot-strapped into an "Oracle". The Oracle class is then responsible for instantiating the auctions, traders and keeping track of which goods have been sold.

4.2.1 Use of JASA

JASA [PMPM06] is an auction simulator developed by Steve Phelps. It allows researchers in agent-based computational economics to run trading simulations using a

number of different auction mechanisms. JASA provides base classes for implementing simple adaptive trading agents and is extensible, so that new auction rules can easily be implemented.

JASA is implemented as a set of extensions to Repast [NCV06] which is an agent based modelling environment, designed for running simulations of a large number of agents. Repast is implemented as a single Java Virtual Machine, in which each agent is represented as an object, with each object having a step function that is called sequentially by the master Repast process. Agents are synchronised through an artificial time mechanism and complete one "round" or action per call of the step function.

Our MDA works on a system of Shouts, which can either be Asks, or Bids. Ask messages come from Sellers, whilst Bids come from Buyers and we made use of JASA in two ways:

- Our CDAAgent class is an extension of JASA's RandomRobinAuction class and has the KContinuousDoubleAuctioneer as a class attribute. The RandomRobinAuction class provides infrastructure needed for managing the auction, such as (de)registering new traders, notification and requesting of bids and asks, etc. The KContinuousDoubleAuctioneer class implements the rules for the CDA, i.e. matching the bids and asks and deciding which shouts should be cleared.
- Our SubTraders, are subclasses of JASA's ZITraderAgent class. The traders "intelligence" level is defined by the strategy they use and we have chosen to use the Zero Intelligence Plus strategy.

4.3 Trading Strategy

JASA provides a choice of two trading strategies: (i) Random Constrained Strategy (often referred to as Zero Intelligence Constrained (ZI-C) in the literature[GS93]), which is a trading strategy in which the agent bids a different random markup on our agent's private value in each auction round; and (ii) Zero Intelligence (ZI) strategy, as outlined in work by Cliff and Bruten [Cli97, CB98]. Agents of this type have a finite trade entitlement, which determines how many units they are able to trade in a given trading period. As our objective was to utilise as simple and widely accepted

strategy as possible, ensuring our focus was on the market structure, we evaluated both but focused on the ZI strategy. In our implementation, we developed ZITraderAgents who become inactive once their intitial trade entitlement is used up, with their trade entitlement being restored at the end of each day.

There is a disadvantage to using ZIP agents, which is that they must know what the current bid or offers are before they can decide whether or not to raise or lower their price. This presents an inherent scalability problem because an agent must track all the transactions for the auctions they are a participant in. However, in any distributed system where there is a common auction system this will be a necessary bottleneck. In our MDA implementation, we found that there was significant overhead in the transfer of messages between Agents in a distributed system, but that these could be mitigated by intelligent message routing, which we considered in detail in Section 4.6.10

4.3.1 Social Welfare in Trading

We considered in Section 2.2 that Social Welfare—the process by which we consider factors beyond moneytary value—was important in our trading process as it provides a mechanism where by traders may determine between two equal bids. Recall if a trader wishes to sell good X to Alice or Bob and both place bids on the same pareto efficient plane, then the trader will use a social welfare function to determine whether to sell to Alice or Bob.

In our implementation,having adopted the ZI traders the focus is on using that strategy and multiple rounds of bidding to try and ensure that all bids received are different. However, if they are not then we have limited information from our CATS data to differentiate buyers and determine which buyer to select. Therefore, in our TraderAgentImpl and ZIPTraderAgentImpl classes we use a simple algorithm to select the buyer based upon historic trades and the number of rounds taken to complete trades (a buyer would be prefered if they have traded with greater quantity and less rounds).

4.4 Algorithms in Pseudo Code

Expressed in pseudo-code, the buyer and seller processes are specified in algorithms 1 and 2.

Algorithm 1 $process_{buyer}$

```
1:  Trader buyTrader = Oracle.get_next_trader()
2:  Bundle bundleToBuy = Oracle.get_buy_bundle()
3:  for good in bundleToBuy do
4:      cda = Oracle.get_cda(good)
5:      while good.is_not_sold() do
6:          good.set_price(bundleToBuy.get_Budget()/bundleToBuy.get_num_goods())
7:          cda.place_bid(buyTrader(good))
8:          while buyTrader.offer(good) = cda.get_counter_offer() do
9:              if good.get_price() = buyTrader.offer(good) then
10:                 cda.clear_auction(good)
11:             end if
12:         end while
13:     end while
14: end for
15: Oracle.return_bundle(bundleToBuy)
16: Oracle.update_success(bundleToBuy, self)
```

The Repast based MDA simulation uses a synchronous step model and at every step (or round) the following three sub-steps are performed sequentially:

1. All Traders check the status of their current bundle. If the bundle has not yet been fully assembled and trades are ongoing, nothing is done. If the bundle has been fulfilled or has failed (the Trader has given up), they acquire a new bundle from the Oracle.

2. All auctions perform one round. A round consists of asking all participating traders to send in a shout. Any matches will be reported to the corresponding traders in the next sub-step.

3. All Traders receive any trade results and the data on fulfilled or failed bundles is collected.

Algorithm 2 $process_{seller}$

```
1: Trader sellTrader = Oracle.get_next_trader()
2: Bundle bundleToSell = Oracle.get_buy_bundle()
3: for good in bundleToSell do
4:     cda = Oracle.get_cda(good)
5:     while good.is_not_bought() do
6:         good.set_min_price(bundleToSell.get_Budget()/bundleToSell.get_num_goods())
7:         cda.place_shout(sellTrader(good))
8:         while sellTrader.offer(good) = cda.get_counter_offer() do
9:             if good.get_price() = sellTrader.offer(good) then
10:                cda.clear_auction(good)
11:            end if
12:        end while
13:    end while
14: end for
15: Oracle.return_bundle(bundleToSell)
16: Oracle.update_success(bundleToSell, self)
```

4.5 Conclusions from the MDA System

From our initial implementation of the MDA system, we learned the following lessons:

- Repast is a good choice for building a uni-processor simulation of a multi-agent environment. It provides an excellent framework and supporting infrastructure which allowed us to quickly implement a lock-step simulation with good tools for tracking auctions, statistics gathering and visual progress monitoring.
- We built a system which was able to complete a large number of trades and assemble bundles in linear time. However, our initial experiments focused on bid-throughput and trading, without evaluating the bundle allocations proximity to optimal. In further experiments, we needed to develop an understanding of the bundles produced.

The MDA system is not without limitations, the most significant being the original implementation was built with a number of the Repast concepts deeply embedded, most significantly the "step" function and therefore the simulation was built on a cycle of (i) execute market, (ii) collect data. This approach has no impact on the market operation, but it massively increases the run time of the overall system because after

each round of the market we sequentially poll all of the agents in the system to collect market data and statistics and in doing so, pause the market. In completing our experiments, we found that the largest amount of elapsed time for the MDA system was taken up with datafile loading, preprocessing and statistics gathering and that the distributed markets did not require very much time (proportionally to overall run time) to execute). Thus our measurements of end-to-end run time contain a large amount of non-auction time (note that a similar issue is present for measurement of combinatorial auctions, so the comparison remains valid) which would potentially confuse a comparison of the core allocation systems.

4.6 MDA Re-factoring in AgentScape

Having developed the MDA simulation in Repast, we wanted to expand the investigation and implement a fully asynchronous mechanism in which the traders and markets were not controlled by an iterative step cycle and could act according to independent market events. We identified a number of limitations in the Repast approach and areas for additional investigation:

- **Speed:** In doing our Repast-MDA experiments, we found that a lot of the end to end run time was being absorbed by the "system" processing of reading files, enumerating over agents and logging results. A much smaller proportion of it was spent in the market mechanism. We wanted to improve the platform so that it was lighter weight and agents could spend more time in the market and auction states and less time reporting performance.
- **Concurrency:** We wanted to introduce just enough concurrency, so that more than one agent could make trading decisions at once.
- **Communication:** Distributed systems embody significant communication overheads, and with the Repast stepped approach, agents spent most of their time waiting to be called. In a distributed system where agents operated continuously, the relative placement of processes and resources would acquire a significance that was not present in the Repast simulation. We needed to understand how much time was spent in communication, synchronisation and trading respectively.

We have therefore re-factored the Repast simulation to constrain the concurrency. We introduced the AgentScape [OB06] platform to build the asynchronous system.

Agent Platforms such as Jade [2] are focused on the development of many agents within a single environment. In this work, we have focused on a comparison of distributed and centralised systems (with the agent based system being the distributed component) and hence the distributed features and support for inter-host migration within AgentScape caused it to be a highly attractive agent platform for us to adopt.

The major benefit of AgentScape is that it allows us to take advantage of the mobility platform it provides to enable agents to migrate to the host they do most processing on, so that the majority of communication messages are local, reducing the impact of additional messaging overhead required to facilitate a distributed architecture.

AgentScape was initiated in early 2000 and consists of a "kernel" layer, on-top of which AgentScape agents can be launched. The system of middleware then facilitates communication, mobility, security, fault tolerance, distributed resource and service management. Initial versions of the AgentScape kernel were implemented in C, but it has since been reimplemented in Java, which is what we used for our developments.

In the Java implementation, an AgentScape kernel is invoked on each host computer, in a separate Java Virtual Machine (JVM). Each kernel environment is registered with the LookupServer, which tracks the location of all AgentScape Kernels and the Agents located within them. AgentScape has a number of key concepts as show in Figure 4.3[3]: Firstly, Agents and Objects are the two basic entities. These reside at a Location. Agents can pass messages between each other through the kernel, typically serialised Java objects to support a form of Remote Method Invocation (RMI). Additionally, AgentScape supports migration of agents from one AgentScape kernel to another. To support this, agents are serialised down the kernel communication path and unpacked at the other end, before being re-invoked. A more detailed description of AgentScape can be found in [OB06]

[2] Jade is available at `http://jade.tilab.com`.
[3] With thanks for Benno J. Overeinder for the figure.

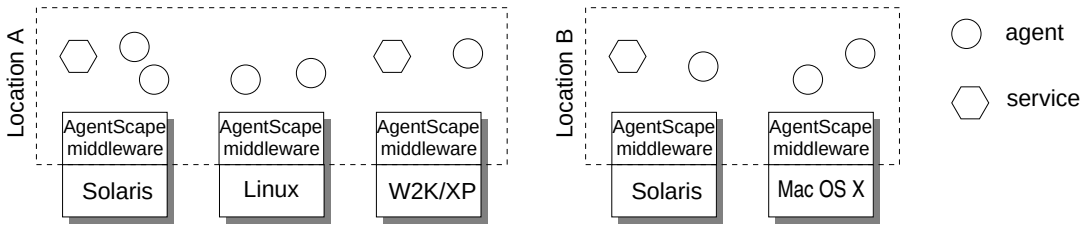

Figure 4.3: Model of AgentScape middleware environment

4.6.1 Using AgentScape for distributed auctions

AgentScape is ideally suited to the development of a distributed environment, such as our auction platform. It contains a number of building blocks (i) agents, which are the active entities of the system, containing the "business logic" for the application. These are hosted at (ii) locations which provide the environment for them to exist and can be found via the lookup service. Finally, AgentScape provides (iii) services which are external software systems hosted by the AgentScape middleware that provide access to things like external web services which are wrapped and presented as Web services, using SOAP/WSDL generated dynamic interfaces.

Agents in AgentScape are defined according to the weak notion of agency (as defined by Genesereth and Ketchpel in [GK94]) and have the following characteristics: (i) Autonomy: agents control their own processes; (ii) Social ability: ability to communicate and cooperate; (iii) Reactiveness: ability to receive information and respond; (iv) Proactiveness: ability to take the initiative. Through the kernel, they may (i) communicate with other agents, (ii) migrate from one location to another location, (iii) create and delete agents and (iv) control service access. We should note that all operations of agents are modulo authorisation and security precautions, a framework for which is provided by the Agentscape middleware e.g., an agent is allowed to start a service if it has the appropriate credentials (ownership, authorization, access to resources, etc.).

Finally, a location is a place in which agents and services can reside. More precisely stated, agents and services are supported by agent servers and (Web) service gateways (respectively) which belong to a location. From the perspective of agents, agent servers give exclusive access to the AgentScape middleware. Similarly, service gateways provide access to services external to AgentScape.

This suite of components provided by the Agentscape middleware made it ideal for our implementation of a distributed version of the MDA.

4.6.2 MDA Distributed Architecture

As described in Section 4.2, the centralised MDA implementation runs the simulation sequentially. In each round, all auctions run in a single thread of execution. However, in a distributed system, all processes that have no direct effect on each other can often be executed in parallel. In the case of the MDA model, each auction could in principle run in parallel which would lead to both improvements in the fluidity of the market (with a greater volume of transactions being completed) as well as general performance improvements, with a significant reduction in the amount of time spent on synchronisation of and data capture from the trader agents. The move to this architecture also allows us to reflect on the actual intended deployment scenario and demonstrate scalability and the impact that physical distribution has on the operation of the MDA.

However, just distributing the application does not automatically mean that it will naturally scale and benefit from running on multiple hosts. The application has to be carefully divided into agents so that these can perform most of their work without requiring too much coordination. If agents can do most of their processing independent of others, a lot of work can be done in parallel.

In order for the results of the distributed MDA to be comparable with the centralised one, the notion of rounds as described at the end of section 4.4 must still be kept. However within each round the autonomy of the agents, which are now able to bid in parallel, lead to improved performance.

In the distributed MDA, the three steps done during each round in the synchronous step model from section 4.4 will still be used, and are executed sequentially. However, within each step, the actions can be done in parallel. In other words, in step 1, all Traders can check their progress simultaneously; in step 2, all auctions can run one round in parallel; and in step 3, all Traders can process the trade results in parallel.

This combination of constrained asynchronous execution has the advantage that the simulation still has the notion of ‘rounds’, and thus producing results that are comparable to the sequential version, while allowing for the maximum amount of concurrency within each round.

In an agent system, typically the work is divided among several agents, which all

perform a part of the work. By using Agentscape to distribute agents over multiple hosts we were able to spread the load of the application across multiple servers.

4.6.3 Re-factoring for asynchronous operation

Our initial Repast based simulation operated on a "clock-tick" basis, with Repast calling a "step" function on every object at every tick, which in turn caused the agent's logic to advance, for example by placing another bid in the market.

However, in the Agentscape model, there are many different agents operating across a number of hosts and this requires an asynchronous approach to operation, with every agent acting independently, therefore we re-factored the framework so that the actual Trader code from our Repast implementation could operate effectively in the new environment.

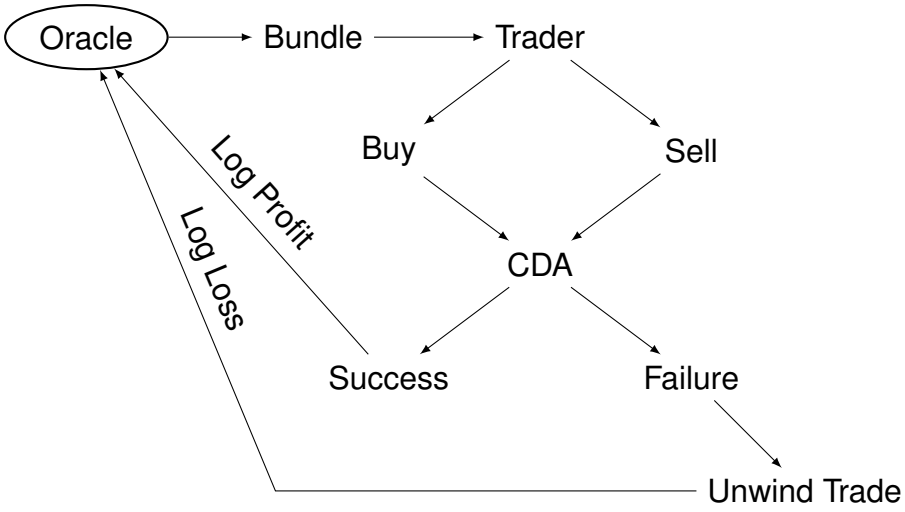

Figure 4.4: Visualisation of MDA bundle flow

We highlight below, for the purposes of ensuring a valid comparison, how the MDA system works in both a synchronous and asynchronous environment.

4.6.4 Initial set-up of the market

The overall flow of bundles is visualised in Figure 4.4 on page 89. The MDA-Agentscape system starts by loading the Agentscape environment in all the relevant machine in-

stances and setting up the initial bundles, traders and CDAs. The first action is to read in the CATs data file, using the BundleReader and this is a centralised process for both the AgentScape and AgentScape-MDA operations.

The BundleReader reads each line of the CATs file containing the list of resources and the funds for the complete bundle. Identical Java objects are used for goods to be bought and sold, with negative quantities indicating a buy request for a good and positive quantities indicating a good for sale. The bundles of requests to buy are created first, with the quantity for each resource being set at -1, subsequently a new sell bundle is created with quantity +1. A Buy request bundle will contain multiple good elements, whilst a sell good "bundle" will only contain a single good. These bundles are then added to the Oracle and the price is calculated. The price for sell bundles are initially set to be the average maximum price available (total budget/number of goods for sale) as the price can then be negotiated down.

4.6.5 Startup and Bootstrap Process

Having read the bundle, the market is started up:

AgentScape: The ZIPTraders are started by the Oracle with an initial bundle (to buy) or good (to sell). They then run for a fixed maximum number of rounds. If no trade has been made in that time frame, the ZIP agent returns the bundle to the Oracle who can make it available to the market at a later point in time.

AgentScape-MDA: In the asynchronous market, the Market Manager mechanism and Oracle are instantiated during initial setup and remain resident whilst the system waits for traders. When a trader joins, it requests a bundle and will then request the location of the relevant CDA market for that good. Having joined the market, the trader can begin trading as soon as the counter trading party becomes available in the CDA.

4.6.6 Auction Clearing Process

Once trading has commenced the auctions will begin to match buy and sell shouts and clear goods.

AgentScape: Buyers enter auctions and place "bid" shouts for the goods that they want in the relevant CDAs. Similarly, sellers place "ask" shouts for the available goods. A parent trader agent is responsible for managing the (dis)assembly of the whole bundle and will dispatch sub-traders to negotiate for a specific good in the relevant auction. Traders will try for up-to 20 rounds to complete the purchase or sale, before giving up and returning the bundle to the Oracle. If an "ask" shout matches the "bid" then the Oracle is informed of the successful trade and the sub-traders close down and report success to their parent trader agent.

When there is a matching buy and sell shout, the auctioneer will clear the auction round. If they do not match, then everyone monitoring the auction is told, and participants may submit a lower/higher ask/bid respectively.

AgentScape-MDA: The asynchronous process is similar, except that whilst the buyers will continue to try for 20 rounds (tests in Gislin Kamda's MSc report showed this to be the optimal number) the sellers will keep selling until they have sold the item. The CDA remains open for as long as the AgentScape platform is running (although there may be no items available to buy).

4.6.7 Returning bundles

In a distributed system of continuous double auctions where each sub-trader was assigned a single good it would be possible for that sub-trader to continue attempting to trade until the end of the market process. However, in the Repast model, as implemented, sub-traders cannot keep their good beyond the end of each "step" and therefore all goods need to be returned to parent traders.

AgentScape: The sub-traders keep their bundles for 20 rounds, which is the length of time each step in the market lasts for. Partial bundles are generally only buy bundles (sell bundles have a single resource only so they either trade completely or not at all).

AgentScape-MDA: Buyers seek to produce whole bundles and when a buyer cannot complete the purchase of all the goods in a bundle after a given number of rounds (20, as in the synchronous market), the Bundle should be returned to the Oracle and the funds associated with it will also be returned and subtracted from the trader's balance. As implemented, we do not penalise the trader for this, beyond reversing the trades and decrementing the profit gained, but this remains an option for future work. Every time the Oracle gets a good to sell from the return process (i.e. one which was purchased in a partial bundle), the Oracle will go through the bundle again and invoke a new seller agent to resell the good.

4.6.8 Market stop condition

AgentScape: The Repast simulation continues to poll all traders until all of the goods have been taken at least once. It then continues to run the market for the same number of ticks again, to give the market time to process any outstanding bids or partial bundles.

AgentScape-MDA: The asynchronous market does not run for a specific number of ticks or time intervals, therefore, the market mechanism is instantiated and runs perpetually. Traders can join and leave the market as required. A CDA continues to operate for the life of the AgentScape platform.

4.6.9 CDA clear conditions

AgentScape: At every Repast step, the CDA attempts to clear the market and determine if there are matching bids.

AgentScape-MDA: At every shout in the market, the CDA will look to see if there are matching bids and attempt to clear the market. The CDA can only begin the process if there is at least one buy and one sell bid in the market (with corresponding buyer and seller trading agents).

Once the auction clears ZIPTraders who monitor but are not successful are notified and they can then submit fresh bids.

4.6.10 Messaging Overhead

Two key issues arose during our implementation of the MDA market in AgentScape:

(i) The high overhead of AgentScape messaging between nodes.

(ii) The inefficiency of the sequential processing model in Repast which meant that the relative concurrency available in Agentscape could not be fully exploited.

In order to make a smooth transition from the existing application to the distributed MDA version, all distributed objects are encapsulated in an agent class. The objects are then accessed by the Agent Proxy, meaning that every single method call between auctions and (sub) traders is translated into a remote method invocation, so method invocations are sent using the AgentScape messaging service. The original application is not really aware of any distribution of the objects, because this is done transparently.

Even though the proxies help to distribute the objects, it makes the entire application a lot slower, because all invocations are now sent over messages. In the sequential MDA, the cost of a method call can be neglected, but for a distributed system this is much more costly, because sending an invocation request and waiting for a message with the result has a much higher latency. This is especially noticeable for a high number of invocations.

Analyzing this message traffic provided insight into which agents communicate the most, and offered a first means to improve performance. It is possible to reduce message traffic by caching method results whenever possible. Another approach is to group method invocations that are often used subsequently, as a single invocation is more efficient than multiple invocations. These optimisations generally do not require restructuring of the application and as such are easy to apply.

Other attempts at reducing the number of messages involves more structural changes. For example, in the MDA application it can be observed that certain agents communicate only with certain others. For example, each resource (sub) trader mostly only communicates with a single auction object. Moving the resource trader to the same node as the auction is executing on, allowing it to perform its work locally to the auction saves a lot of network based messaging.

Another issue that involves a lot of messaging is polling. In the sequential MDA, traders poll the auctions for results of their sub-traders on every round. A notification scheme, where the auction notifies the traders whenever results are available removes a significant proportion of messages.

4.6.11 Synchronisation Coordination Overhead

In order to ensure comparability of results from both the centralised and decentralised systems both use a single manager for keeping track of rounds. On each round, every agent runs its part of the round, after which it waits for the next round.

This means that in the distributed MDA a large amount of time is spent waiting. Even though auctions can run in parallel, they cannot continue working on a new round until all others are finished with the previous round.

Having to synchronise is not problematic if the costs of doing so are relatively small compared to the amount of work that has to be done by the agents. In this application, however, the average time for an auction to process shouts is less than the time it takes to send and receive a message. For example, when running an auction with a lot of agents, each time a bid is received a message is sent to the agents participating informing them of the new bid and giving them the opportunity to update theirs.

A larger grain size (i.e. if the messages contained more information/work) would reduce the impact of the messaging. This could be obtained by allowing traders and auctions to perform multiple steps at once. However, in the current design traders check the status of their bids in all of the auctions they participate in for every round. This operation allows them to maintain accurate control of funds, but it restricts the auctions and traders to performing only a single round at a time.

If the amount of work done by each individual agent is what demands most of the processing time, then distribution would be more successful. However, in the MDA simulation the cost comes from having to perform work for many individual agents. All agents have to be coordinated, though each individual agent performs relatively little work.

4.6.12 Evaluation of Agentscape-MDA

The software engineering challenge we faced in our Agentscape-MDA system lies in how to scale that demonstrator up into a system comprising many more agents running over multiple machines. We took an incremental approach, re-factoring the MDA system into a distributed system, but it is clear that we did not go far enough in our work because it seems that whilst increasing the size of the agent platform does not impact on the market dynamic in the system, it does impact massively on the runtime.

Testing Objectives

In our experiments there are two variables. Firstly, we can expand the number of servers running the Agentscape environment, to test the effect of the inter-environment communication. Secondly we have run multiple datasets on both environments, to determine the runtime performance of the systems relative to each other.

Our objectives in testing were to consider the impact of the overhead imposed by the communication between nodes, which we can demonstrate by increasing the number of nodes in the system and monitoring runtimes and market throughput.

We would anticipate that increasing the number of processor nodes would increase the CPU available for the system, but that adding more nodes will distribute the problem more sparsely and hence increase the amount of inter-node communication.

The second objective was to run a number of simulations using the L2 dataset, to compare both the overall runtimes and market throughput of the Agentscape and Repast simulations.

In our AgentScape simulations the experiments used 400 trading agents.

Initial AgentScape Results

Our AgentScape simulations were run on both a single computer and on multiple computers. When communicating locally, the inter-agent method calls pass directly be-

tween Java objects. When the agent is remote, AgentScape uses the custom RMI mechanism to faciliate communication between agents running on different AgentScape platforms.

(Bid Throughput is defined as the total number of bids traded in the market simulation.)

Run A — 1,2 and 4 AgentScape Environments:

No of Environments	Runtime (min:sec)	Bid Throughput
1	20:46	18469
2	50:53	17015
4	74:08	16253

Run B — 2,4,8,16 AgentScape Environments:

No of Environments	Runtime (min:sec)	Bid Throughput
2	43:52	18469
4	61:18	19202
8	72:33	17368
16	81:26	16232

Run C — a single Repast Environment

No of Environments	Runtime (min:sec)	Bid Throughput
1	0:59	18354

We are also able to show equivalence, across both the four AgentScape environments and the Repast environment, as the number of bundles traded, completed and unfulfilled across the market are consistent for all iterations of the experiments, as shown in 4.5.

It can be seen from these tables that all experiments produce similar market throughput. Additionally, all experiments run to completion within the same number of "ticks" (clock cycles).

The key difference between all our runs is that the runtime in AgentScape is much greater than the Repast runtime, and that the AgentScape runtime is greatly extended as the number of processors increases.

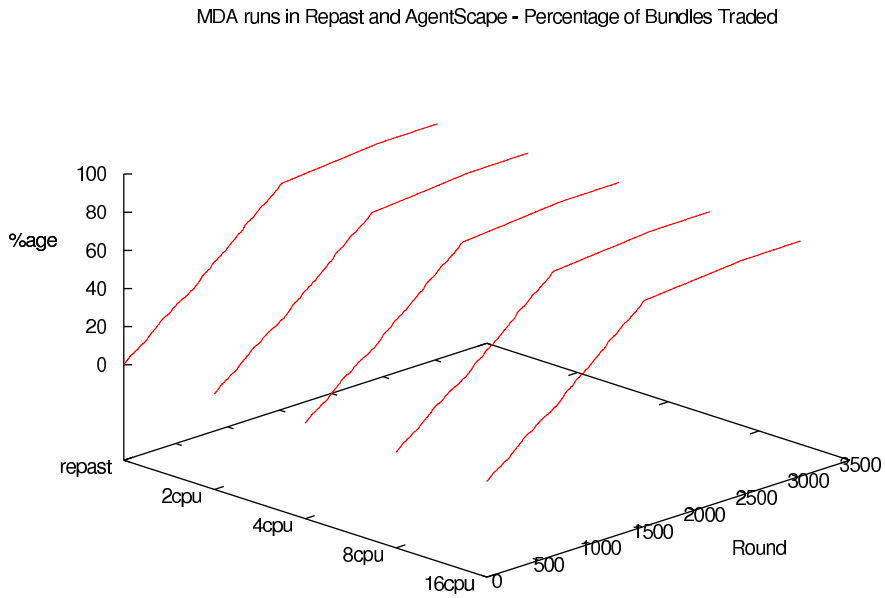

Figure 4.5: MDA Bundles Traded - Repast and AgentScape

AgentScape Implementation Evaluation

MAS software development is characteristically evolutionary and a common starting point is a proof-of-concept system running on a single machine utilizing an agent platform or even a simulation framework. The software engineering challenge lies in how to scale that demonstrator up into a system comprising many more agents running over multiple machines. A large-budget solution might throw away the demonstrator and rewrite from the ground up, but there are many risks with this approach as well as high costs. An alternative approach, that is the subject of this paper, is to refactor the demonstrator into a large-scale system, taking advantage of the incremental nature of the changes applied and regression testing to create confidence in the process and the outcome.

We have pursued this approach and developed a system to support large numbers of agents participating in large numbers of auctions on geographically distributed machines. However, from our initial experiments, it seems that whilst increasing the size of the agent platform does not impact on the market dynamic in the system, it does impact massively on the runtime.

This dramatic increase in runtimes is due to the large number of messages that are passed between trader agents, all of which need to be serialised and sent through the AgentScape message system. The amount of communication that is required to keep

the agents busy is simply too much with respect to the amount work that each individual agent needs to perform. The CPUs sit idle for too long while waiting for messages.

In this particular simulation, it would have been better to utilize multiple CPUs on a shared-memory memory machine (such as a single multi-core machine), thereby reducing the latency from message sending whilst still distributing the work over multiple processing units.

However, the choice to constrain the asynchronous nature of the distributed simulation by keeping the notion of rounds, did make it possible to compare the results with the centralized simulation. Unfortunately, this choice also meant that a lot of synchronization was necessary among traders and auctions, which lowered the overall performance. A fully asynchronous simulation would probably show better performance.

Another lesson learned was that placing agents on hosts must be done with care. Placing agents that communicate a lot close together (i.e., on the same host) improves the performance of the simulation considerably. For example, sub-traders, that try to acquire a single resource at an auction on behalf of a trader, should be placed on the same host as the auction. Unfortunately, there are also agents that cannot be easily placed together. For example, the traders themselves cannot be placed at auction's location, as the trader has to interact with many different auctions to acquire a bundle of resources, and auctions are best placed at different hosts to take advantage of parallellism.

In conclusion, it has proved to be difficult to optimise the speed of the MDA market simulation by distributing over multiple hosts, while still respecting the synchronisation contraints from the original Repast simulation. Synchronising agents is very time consuming due to the amount of messaging involved.

An alternative approach to reduce the communication overhead, without the need to change the structure and algorithm of the simulation, is to use a shared address space for all agents. For example, running the simulation with multiple threads on a single (multi-core) host would not require RMI communications but would still benefit from running auctions and traders in parallel.

4.7 Conclusion - the MDA System

The MDA system has been built to provide a distributed market mechanism for performing resource allocation. It takes input from CATs generated test datafiles, extrapolates them and uses intelligent agents to trade in a series of continuous double auctions to complete a resource allocation.

As well as using an approach of distributed decision making to resource allocation, we have implemented MDA using both the single-host repast simulator and a multi-host Agentscape based simulator and shown that there are significant engineering challenges to be considered when running a multi-agent system with a high volume of transactions in a networked environment.

However, the use of a distributed market mechanism has provided us with a reliable way to simulate resource allocation in an environment with no centralised single point of control and that allows us to begin looking at the question of which resource management technique is the most appropriate for any given situation and how the outcomes produced by centralised and decentralised systems compare.

Chapter 5

Experimental Approach

In evaluating our experimental approach we need to consider a number of areas. Firstly, we must address and understand the differences and similarities in the centralised and decentralised system and subsequently confirm that there is a sound basis for comparison. With that established we then identify four metrics which provide measures for comparison. Next, we consider the structure of our experiments and the make up and selection of test data, before concluding with a discussion on the structural integrity and validity of the two software algorithms, CASS and MDA.

Having evaluated the approach, we turn to the actual experiments and describe the specific simulations completed and results captured.

5.1 Comparison Problem

We have previously discussed the problem of comparing two systems—one centralised, one decentralised, but it is beneficial to recap on why the comparison is both valid and useful in the context of discovering the most appropriate resource allocation mechanism for a given set of circumstances.

We achieved our comparison using a common set of inputs (from CATS datafiles) and then producing the same outputs—a list of which bidders had 'won' their goods and the corresponding costs and values.

Some will argue that a centralised system is not solving the same problem as a decentralised system but we disagree. Whilst they do use entirely different methods, they begin with the same data and objectives and end with outcomes that are objectively the same, in that an allocation formed from preferences is completed.

It is accepted that to achieve a Pareto optimal allocation you must have complete preference information for all market participants, which is the situation with a centralised mechanism. With a distributed system, participants will only release as much preference information as is necessary. In this scenario, the market will achieve a set of outcomes each of which is Pareto efficient (no goods can be reallocated to make anyone better off, without disadvantaging others) however, it is impossible to know which, if any of these are the Pareto optimal solution. However, as discussed in Section 3.2.1 (page 58) we do reach a point on a Pareto frontier and in a market mechanism, all further optimisations must optimise on the basis of social welfare.

5.2 Factors for comparison

We have identified the following factors for comparison:

Hardness: In "Combinatorial Auctions" [LBS05], KLB presents "Gross hardness" figures for a range of CATs distributions and we reproduce that graph, of runtime orders of magnitude, in Figure 6.1. We are able to perform a similar analysis of our own data, along with that from KLB's CPLEX and CASS simulators.

Time: The Combinatorial Auction runs as a single-shot process and has a defined end point. We are therefore able to easily determine the amount of actual time (which can be scaled relative to different CPU speeds) taken to compute solutions. In the MDA, run time measurement happens differently, because the market mechanism runs continually, however, there is a specific amount of time between starting with a fresh dataset of goods and bundles in the market and the point at which the traders in the market will have either (a) successfully completed their bundles or (b) concluded that they will not be successful and will have "given up". In our MDA simulations, we looked at the completion rates after 500 rounds, as after that point we determined that the market was largely inactive.

Financial: Each bundle has an assigned value and the object of the Combinatorial Auction is to maximise the revenue earned by the sellers through the sale of bundles which obtain the highest set of purchase prices. For the MDA, we extrapolate the CATS bundle prices to give individual goods a buy/sell price (by uniformly splitting the available overall price) and so for each bundle we can measure the distance between the total sale price of all goods sold (potentially for sets of incomplete bundles) and the total available budget. This allows us to determine the "under-spend" incurred through only purchasing partial bundles.

Efficiency: Building on our discussion of the behavioural properties of computation markets in Section 2.2 (page 50), in which we considered that a combination of pareto optimality and social welfare maximisation defined "efficiency" we shall specifically define market efficiency for this analysis as being the total value of trades undertaken. In the CASS environment many of the bids are not met because they do not represent a Pareto optimal solution, however, that also means that there is value available to sellers which is not "unlocked". Potentially it would be possible to generate more value for sellers by complementing the complete bundles sold as part of a Pareto optimal outcome with the sale of goods from partial bundles, thus giving us a social welfare benefit in our "value of trades" measurement.

Completion: In the Combinatorial Auction, the mechanism is such that it finds the optimum solution for complete bundles. Bidders either receive all of their requested goods, or none of them. The MDA on the other hand is a linear process and bidders buy their goods sequentially with no certainty that they can complete the bundle. Therefore at the specific point in time when the mechanism is stopped there will be a mixture of partial and complete bundles. In our model, each good within the bundle has equal value and no preference is attached to any specific good. Therefore we propose to measure completion as a percentage score of the number of goods out of the total desired bundle obtained at the point the market snapshot is taken.

Satisfaction: We have presented criteria which allow us to make quantitative comparisons of the solutions we have computed from our CASS and MDA solvers. In order to provide a statistical comparison it would be helpful to have a single, combined metric by which we can measure the relative performance of the different approaches. We have defined this parameter as "satisfaction" and consider it to be the product of underspend and the completion ratio for the bundle. Expressed algorithmically, we say:

$$\$satisfaction = \$underSpend * \$completionPercentage$$

The rationale for this is as follows:

- If I complete my bundle, I am happy. If I purchase it cheaply, I will be happier than if it has a high cost. If I get 100 percent completion for a high underspend value, then I will have a high satisfaction rating.
- If I get 50 percent of the goods desired for lots of money, I will have a very low underspend value which will be divided by 50 percent, giving me a very low satisfaction ratio.
- If I get 50 percent of goods desired for little money, I will be reasonably satisfied, so my high underspend value will be divided by two thus taking the downside of an incomplete bundle into account.

Through the use of these variables we will be able to develop a quantitative analysis of the relative merits of the centralised and decentralised approaches and we shall explore these themes in more detail when considering the results of our experiments.

5.3 Construction of Experiments

We used the Combinatorial Auction Test Suite (CATS) for the generation of test data, and in particular, data published by Kevin Leyton-Brown (KLB) [LBPS00], as the basis for our comparison.

KLB produced "empirical hardness-models" as researchers find it very useful to be able to predict how long an algorithm will take to solve a particular problem instance. This will allow the user to decide how to allocate computation resources and whether a complete or approximate algorithm should be used.

KLB published test data for a number of different hardness models from two different sources. The first, published along with the initial wave of algorithms for solving the Winner Determination Problem (WDP) are known as L1 (and L2) through L7. These were criticised in several ways, perhaps most of all for lacking economic justification, a significant criticism because a combinatorial auction is a weighted set packing problem and if the data upon which algorithms lacks any connection to the combinatorial

auction domain it is reasonable to ask whether the algorithms also have any connection.

On the other hand, these datasets do have published work associated with them and given that a core principle of our comparison is that we are able to compare our results against work by others, we are keen to include them.

Therefore, we selected the L2 dataset, produced using the Weighted Random distribution from Sandholm [1999], which chooses a number of goods g from [1,m] and assigns the bid a price drawn uniformly from [0,1]. We made the selection based on recommendations by KLB in [LBNS02] in which he considered the percentage of dominant bids generated over a number of runs and was able to identify which distributions were trivially easy (L1 and L5). L2 was less trivial and hence we adopted it.

Concerns the original datasets (L1 to L5) were trivially easy to solve lead KLB to develop a new version of his CATS systems which consisted of a number of distributions paths, regions, arbitrary, matching and scheduling. These seek to model real world situations, for example Paths models an auction of transportation links between cities, whilst Regions models an auction of real estate. We choose to work with the Scheduling distribution as that models a distributed job-shop scheduling domain, with bidders requesting an XOR'ed set of resource-time slots to satisfy their specific deadlines. The scheduling distribution, therefore, had the most in common with our distributed market problem. We were also interested to determine how our model fared with a harder distribution and so we have conducted limited experiments with the Arbitrary dataset, but we lacked sufficient computing power to develop a complete set of a results.

5.4 Structure of Test Data

We also need to consider how big our test dataset should be. Most researchers fix their problems in terms of the number goods and the number of bids and indeed, problem size is a reasonably well understood parameter of hardness. However, KLB showed that whilst it was easy to fix the number of goods, it is much more difficult to keep the number of bids constant. One of the main problems is the need to ensure that there are

no dominant bids in the test data. KLB defines the concept of dominant bids as being:

> Bid i is dominated by Bid j if the goods requested by i are the same or a superset of goods requested by j and the price-offered by i is smaller or equal to the price offered by j.[LBNS02]

One can see that if a set of bids contains those which are offering less money for more goods, then they should be automatically excluded as they do not contribute to the hardness of the dataset in any way. Indeed, most solvers (e.g. CPLEX) employ a polynomial-time reprocessing step to filter them out first.

Each of the test datasets is generated as a bid-graph, and KLB built the distributions using 30 features, which group into 5 broad feature areas: (i) Bid and Good node statistical features for the links between bid and good nodes, (ii) Bid Graph structural features, such as edge density, path length, (iii) Linear Programming based features (these make it harder for the CPLEX solver), (iv) Price based features, and (v) Problem Size features.

We choose a fixed problem size of 256 goods, 1000 non-dominated bids, with 500 instances of the problem as these were pre-produced by KLB along with associated results computations. Based on KLB's results the "L2" and "Scheduling" datasets looked feasible but non-trivial to compute using the CASS and MDA systems. We have also conducted limited investigation with the "Arbitrary" dataset, but this is a much harder problem set to solve, containing problem instances between 2 and 5 orders of magnitude harder, as shown in Figure 6.1.

For the majority of graphs therefore, we plotted 2000 test instances, 500 each for the L2 and Scheduling distributions for both CASS and MDA systems. Each test instance is a bid for a bundle of goods and each of these bundles has characteristics which we examined, typically through aggregate measures (such as standard deviation and variance).

5.5 Calibration of timings

During the course of the PhD a number of different computer systems were used to test programs and calculate the solutions and many early results were computed using the Oxford National Grid Service Cluster a set of 64 dual CPU Intel Xeon 3.06 GHz server nodes and individual experiments were computed on two departmental workstations, which contained high spec Xeon processors.

Subsequently all calculations were recomputed using the Aquilla cluster at University of Bath, which provided 50 quad CPU Intel Xeon 2.8 GHz server nodes and graphs shown in Section 6.3 are all computed consistently on this platform.

In order to calibrate early results to the final "aquilla" computations, timings were converted via multiplication to the number of clock-ticks executed on the various CPU models (according to CPU specification), such that the comparison was consistent regardless of speed of actual execution.

5.6 Verification of accuracy of CASS

KLB initially used the commercially available CPLEX solver to determine his solutions, but then went on to propose his own solver, the Combinatorial Auction Solution Solver (CASS). CASS is provably optimal: It finds a solution that maximises the social welfare for a particular scenario and its answer can be compared directly to other algorithms, such as BidTree [SSGL05] and CPLEX.

As we know from our earlier discussion on Pareto optimality and social welfare optimisation, it is possible that a frontier might exist with a number of complementary solutions (i.e.: that yield the same social welfare with different bids). In the published literature there appears to be an implicit assumption that the solutions that CASS produces are identical to the ones that, say, CPLEX or BidTree produced. Whilst there can, of course, be only one answer, with respect to the value of the solution, a Pareto frontier represents a number of different combinations of goods and there does not appear to be any published work comparing the different combinations of outcomes for centralised solvers.

In email correspondence with KLB (August 2008), he said

> "CASS is provably optimal, so are BidTree and CPLEX. It's possible that there will exist solutions that tie (i.e., that yield the same Social Welfare with different bids); in this case, the different methods could find different sets of bids. I've done experiments confirming that CASS really does find the optimum, checking against CPLEX (I've never been given code for BidTree or CABOB). So have others, e.g., Craig Boutilier. As far as I know these results haven't been published."

It is unfortunate that there is no published analysis of the possible different solution sets, however on the basis that KLB's work has been widely scrutinised, published and produces the same fiscal (if not component) output as other solvers, such as CPLEX, we feel confident that it provides an excellent basis for comparing the MDA solution.

One further disadvantage for the CASS is that whilst KLB has published raw test data and his run times and optimum solution values (i.e. the amount CASS spends) for the bid problems he presents, he has not published the solutions or details on which bids won the process. CASS is deterministic and we assume, that our execution of the CASS algorithm with the same data gives us the same optimal value (which we have checked) from the same winning bundles (which we cannot check), but in faster time (our CPUs are newer and run faster, so this is to be expected).

5.7 Conclusions

We have outlined the two market mechanisms that we wish to compare and given a detailed overview as to how we will complete the comparison. Importantly, we have identified our metrics hardness, financial, completion, efficiency, time and satisfaction which allow us to highlight the relative performance of the two mechanisms.

Having developed a comparison framework, we then considered the test data, selecting the L2 and Scheduling datasets. It is important to ensure that neither the test data nor the models shall trivialise the problem. We examined MDA in detail in Chapter 4 and here we have considered problem of dominant bids in the test data and the accuracy of the CASS system.

Chapter 6

Experimental Results

6.1 Introduction

Through the use of our key metrics (i) Hardness, (ii) Time, (iii) Financial Performance, (iv) Efficiency, (v) Completion and (vi) Satisfaction we are able to evaluate the different facets of market performance delivered by the two algorithms, CASS and MDA, in an objective manner. We have already identified in our earlier discussion that both algorithms are useful and perform well in different circumstances. Through drawing a comparison we are able to identify the characteristics they exhibit that allow us to give recommendations, based on empirical research, as to when the use of each mechanism is more appropriate.

In examining the data we were able to observe patterns and characteristics that we felt were unusual or inconsistent, but typically related to our preconceptions about the "hardness" of the distribution sample set. Therefore, to ground our results in context, we begin with an analysis of KLB's previously published results using the CPLEX combinatorial auction solver software.

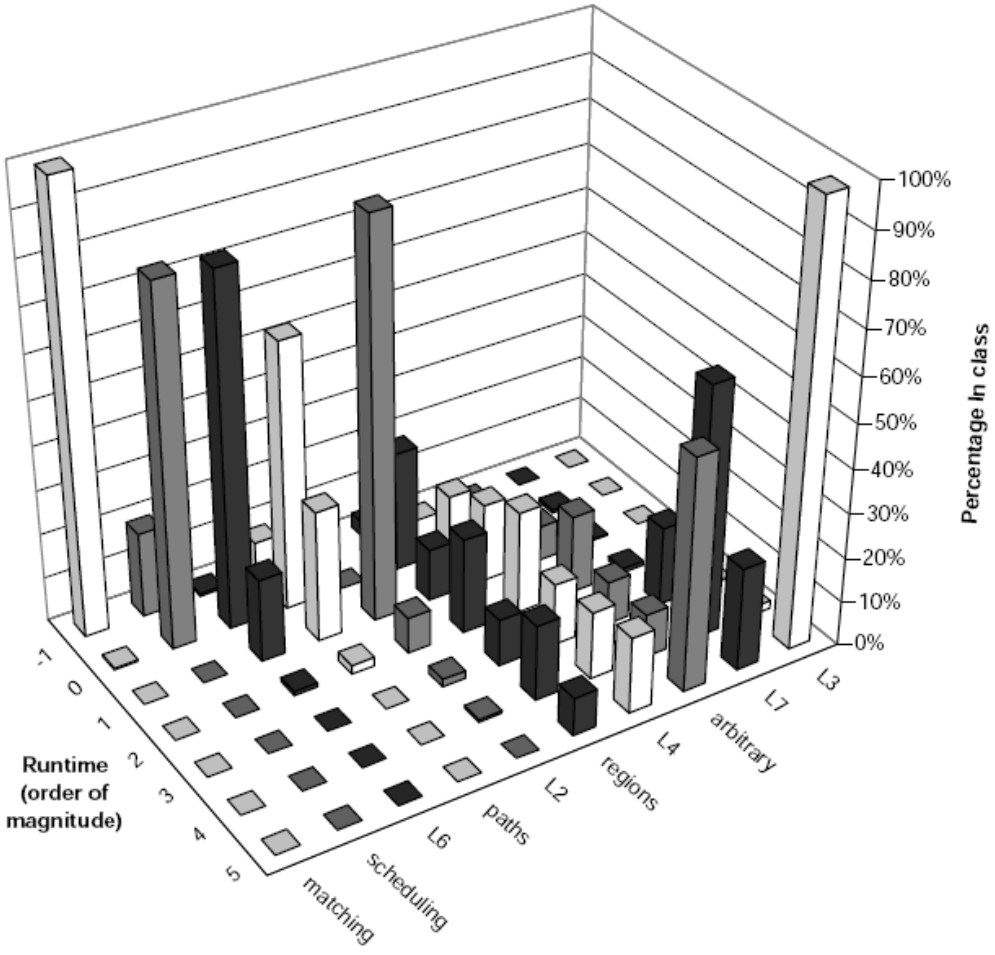

Figure 6.1: Gross Hardness, 1000 Bids/256 Goods, Graph by KLB

6.2 Examining Hardness

In "Combinatorial Auctions" [LBS05], KLB presents "Gross hardness" figures for a range of CATs distributions and we reproduce that graph in Figure 6.1. This figure shows the results of the same 500 runs for each distribution on problems with 256 goods and 1,000 non dominated bids, indicating the number of instances with the same order-of-magnitude runtime—that is $\log_{10}$(runtime). These experiments were run on a cluster of Pentium III Xeon 550-Mhz machines and took over a CPU-year to gather. Many of the distributions are shown to be easy for the CPLEX solver and the first reaction might be to suggest that scheduling, with approximately 80% of outputs having runtime that is one class faster than the L2 results shown, would be to suggest that L2 is "harder" for CPLEX, and indeed, KLB confirms this:

> "We can see that several of the CATS distributions are quite easy for CPLEX, and that others vary from easy to hard. It is interesting that most distributions had instances that varied in hardness by several orders of magnitude, despite the fact that all instances had the same problem size."

The test samples developed by KLB (matching, scheduling, paths, regions, arbitrary) are considered realistic, so KLB goes on to consider whether the distributions that are considered "easy" can be made harder and he concludes in [LBNS06] that this is indeed possible.

However, problem hardness is not the only interesting property for a distribution and KLB goes onto say

> ...this evidence suggests that realistic bidding patterns may often lead to much more tractable winner determination problems than the hardest unrealistic distributions such as Uniform/L3. This is good news for those who hope to run practical combinatorial auctions.

We agree with this analysis—and have gone further to suggest that real world problems have different characteristics to the theoretical models typically used in combinatorial auction analysis. But KLB's work (Figure 6.1) might lead the casual reader to assume that a problem which is hard for one solver (in this case CPLEX) is also hard for others that try to solve the same problems in the same manner—we even went so far in previous work [GP07] as to state that

> In terms of difficulty, KLB has shown that the majority of L2 problems have a runtime of approximately 1 order of magnitude greater than Scheduling, when computed using KLB's CASS solver. We therefore consider L2 to be more difficult to solve.

In trying to understand our results we looked to see how our runtimes compared to KLBs CASS experiments, but first, we realised the need to address a more fundamental question—how do KLB's CASS results compare to his CPLEX results?

Firstly, we wish to deal with the outcomes of CASS and CPLEX. Both algorithms produce identical optimum solution values (the data is public and verification is trivial),

and although we cannot verify which bundles are selected (that data is not published) we assume it is identical, given the identical solution values.

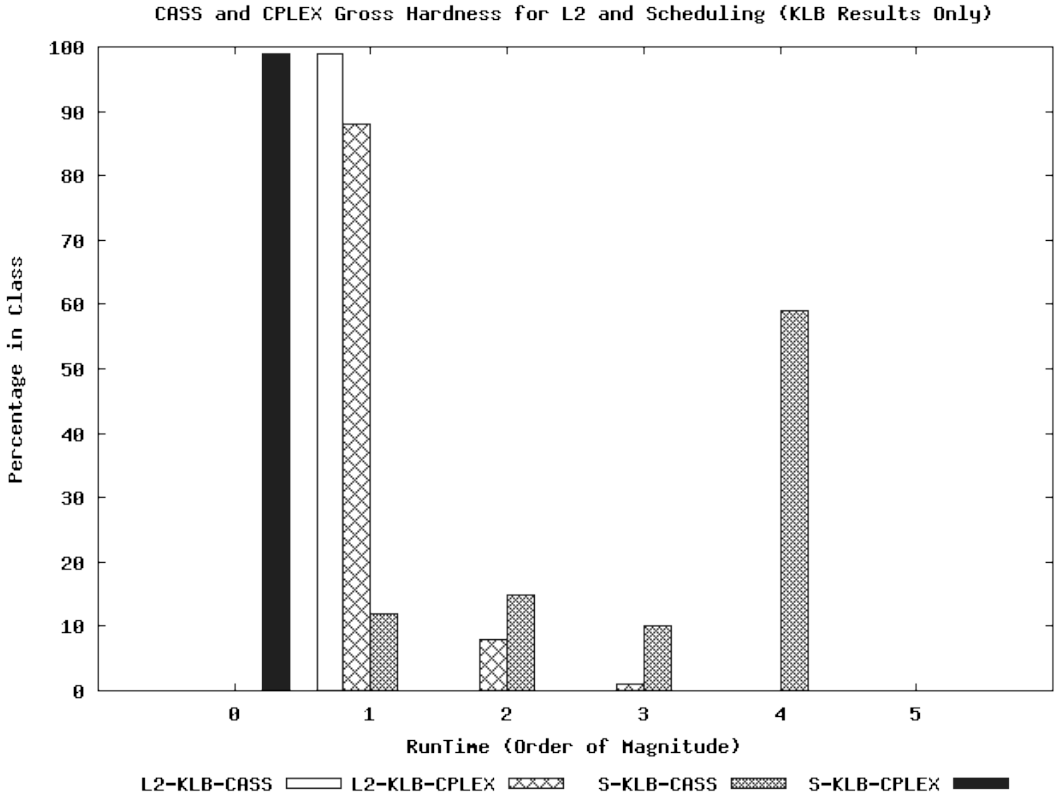

Figure 6.2: CASS and CPLEX Gross Hardness for L2 and Scheduling (KLB Results Only)

In Figure 6.2 we extract KLB's values for the runtime of the L2 and Scheduling datasets using CPLEX and CASS algorithms. A visual comparison of this data in 3D form, as shown in Figure 6.3, shows that the CPLEX figures provide the same order of magnitude as previously illustrated in Figure 6.1. This graph shows us that CPLEX finds the Scheduling distribution (S-KLB-CPLEX) very easy, with all results sitting within magnitude order zero. L2 (L2-K-CPLEX) is considered a little harder for CPLEX, with nearly 90% of solutions sitting within order 1 and 10% in order 2. However, when we compare KLB's runtime results for the CASS algorithm they are different and we see the CASS runtime for the Scheduling distribution (S-KLB-CASS) is spread widely across orders 1,2,3 and 4 (whereas 100% of the results are in order 1 for CPLEX) suggesting that CASS finds the Scheduling distribution much harder to solve than CPLEX, whilst, for L2 (L2-K-CASS), CASS finds it marginally easier to solve than CPLEX, with 100% of results found in order 1 (CPLEX has only 80% of results in order 1).

The difference in these results, such that we might consider them to produce "opposite" results suggests the intriguing notion that different algorithms and approaches will find different aspects of a distribution difficult. CPLEX employs a number of complex preprocessing steps before initiating a branch and bound search whilst CASS uses an approach of (i) structuring the search space to eliminate conflicting bids, (ii) pruning the space to eliminate parts of the bid-tree that will not achieve more revenue than the current best allocation, through use of an overestimate function and (iii) ordering heuristics to capitalise on the structured approach and improve performance.

If we add in our MDA runtime results we can see that MDA, as one might expect, demonstrates different characteristics. Figure 6.3 shows us the runtime, again using the same $\log_{10}$(runtime) value, but now with the four datasets (CPLEX by KLB, CASS by KLB and ourselves, MDA by ourselves) across both L2 and Scheduling distributions for comparison.

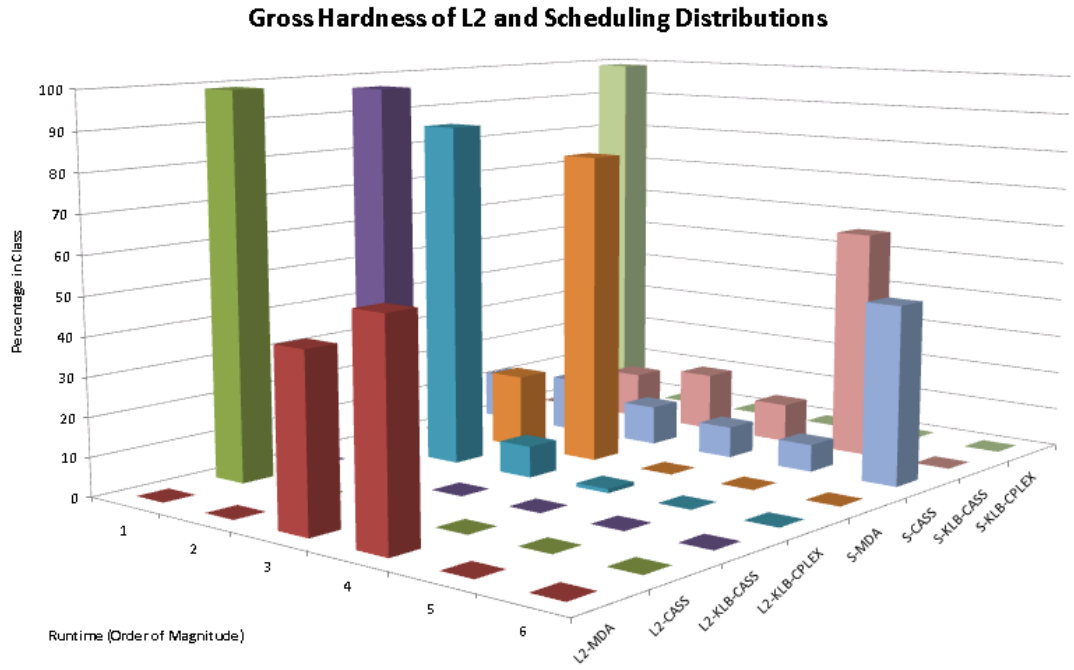

Figure 6.3: Gross Hardness of L2 and Scheduling Distributions

We can draw several insights from this data. Firstly, our execution of the CASS environment is typically an order of magnitude faster than that of KLB, which is simply an artifact of processor speed improvements. Secondly, we see that MDA finds L2 marginally harder than Scheduling, a pattern that is similar to CPLEX, but different to CASS which as discussed, has a wide distribution of runtime orders of magnitude for the CASS-Scheduling results.

6.3 Time

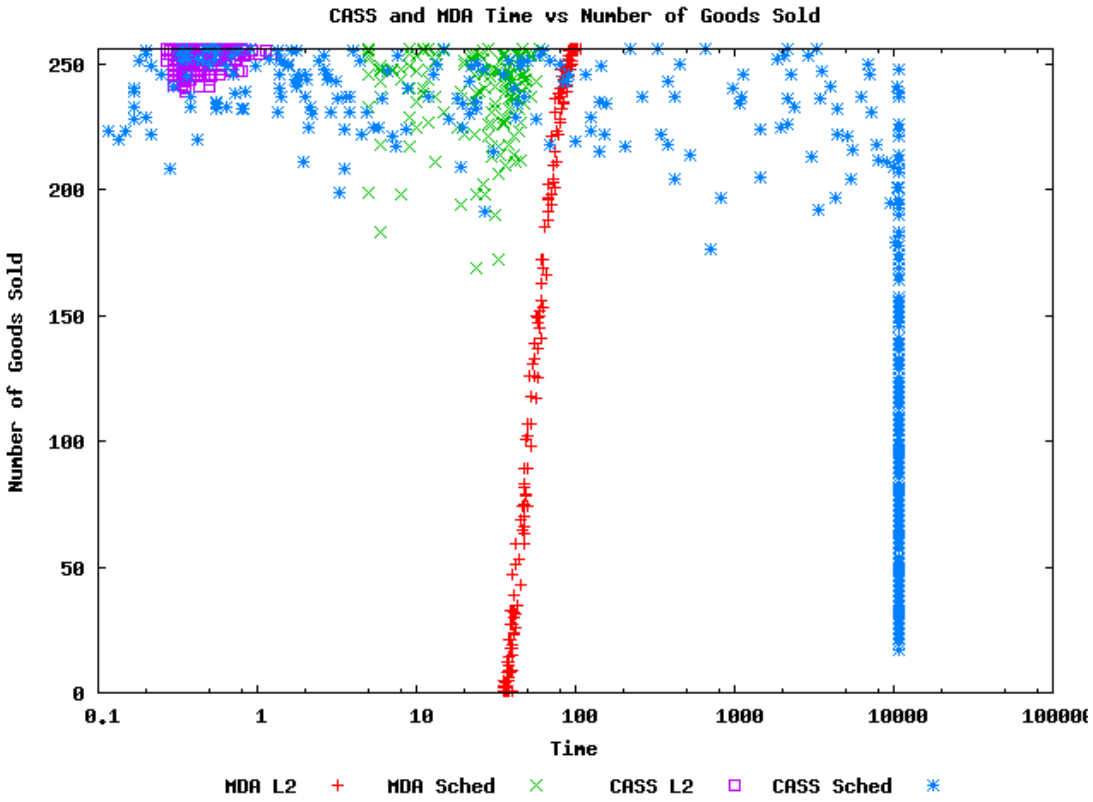

Figure 6.4: CASS and MDA Time vs Number of Goods Sold

With some understanding in place of the performance characteristics of the different distributions and their relative runtimes we turn to evaluate actual values, shown in Figure 6.4. This graph shows us the number of goods sold in the market, along with the time taken to do so. Time is presented in seconds and all experiments were conducted on the same hardware and operating system environment to ensure comparability.

The number of goods sold has been computed by calculating the composition of the bundles which form part of the final solutions generated by each algorithm. There is a maximum of 256 goods available. CASS runs either to completion, or until 10800 seconds, our "wall time", whilst MDA runs for 500 rounds and we record its final position at the 500 round mark.

Figure 6.4 has a logarithmic time scale of actual execution time, which shows us that for the MDA algorithm the amount of time taken to complete 500 rounds is very much a consistent result for all problems as the time elapsed is due to the market process, rather than the size or complexity of the problem. Much more variable is the number

of goods sold showing us that it is the structure of the problem that defines whether or not we can obtain a high level of goods sold. For CASS, figure 6.4 illuminates figure 6.3 showing that all of the L2 instances are solved very quickly, whilst most of the Scheduling instances are actually unsolved and, for those which are, there is a large variance on the number of goods used in the solutions.

Looking at runtimes in isolation has told us that our MDA system provides more predictable run times (experimentation showed us that 500 rounds is a sensible number and extending the runtime does not improve the solution) but the CASS algorithm is much more variable. For some problems L2 is faster and uses a greater number of goods. We need to consider these results in the context of the value and efficiency of the result, to draw further conclusions.

6.4 Financial

In a Combinatorial Auction (implemented as CASS) each bundle has an assigned value and the objective is to maximise the revenue earned by the sellers through selecting combinations of goods into bundles that obtain the highest set of purchase prices.

For a Distributed Auction (implemented as MDA), buyers are in control of making purchases and will tend toward their lowest price. (We extrapolate the CATS bundle prices to give individual goods a buy/sell price through uniformly splitting the available overall price) and so for each bundle we can measure the difference between the total sale price of all goods sold (potentially for sets of incomplete bundles) and the total available budget. This allows us to calculate the total spend for the bundles completed, defined formally as the sum of buying prices for all goods bought in a bundle. Figure 6.5 can be compared with figure 6.4 and represents the total spend per problem for the same set of data instances and number of goods sold. Unfortunately, it is very difficult to meaningfully represent the three values, Spend, Time and Number of Goods Sold in a single graph, however, we can see that spend increases with the number of goods purchases and there are no instances in either algorithm where the entire budget is being spent on a small number of goods. This creates some confidence in the correct functioning of the algorithms.

Comparing MDA and CASS results, we see that MDA is spending less on goods (over

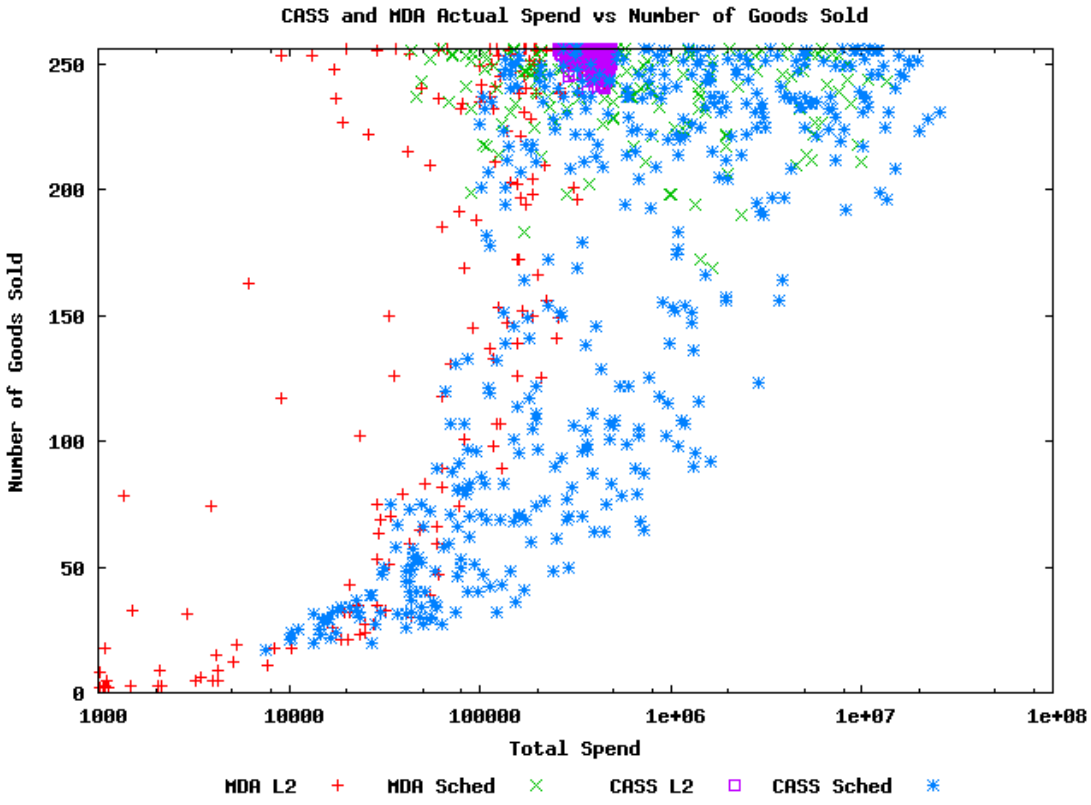

Figure 6.5: CASS and MDA Actual Spend vs Number of Goods Sold

all and per good) than CASS, particularly for the Scheduling distribution, but also for the L2 distribution. This again is evidence of the two different strategies (markets maximise for buyers, auctions for sellers) working effectively.

CASS utilises 100% of the available funds for a bundle because it simply looks for bundle offers that will maximise the gross revenue to the sellers. However, there are some problems where the optimum sellers revenue can be obtained without having to purchase all of the available goods. We can see this in Figure 6.6, with the CASS algorithm producing a high level of good utilisation for L2, but quite poor good utilisation for the Scheduling results which hit the wall time (approximately 30%). MDA has higher good utilisation for Scheduling, but fares worse for L2, consistent with the results we saw in Figure 6.5.

We know that CASS will always spend 100% of the available budget for a bid, so considering just the MDA results, we looked to see what percentage of the budget had been spent on purchasing goods. We would consider an algorithm to be very successful if lots of goods are bought for a low budgetary amount. Figure 6.7 shows that MDA is

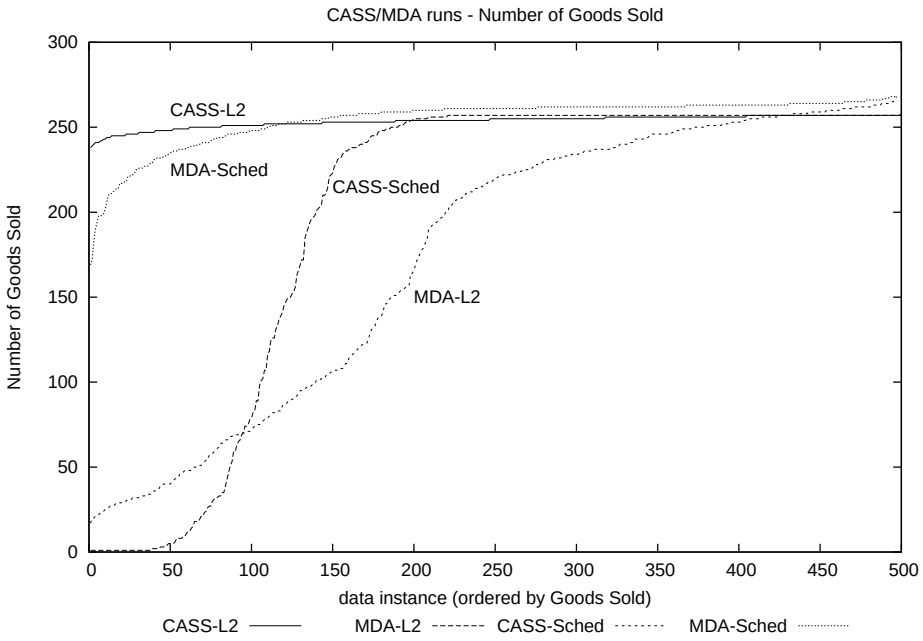

Figure 6.6: CASS/MDA runs - Number of Goods Sold

particularly successful for the Scheduling distribution with most of the test instances having solutions using over 80% of available goods, for less than 35% of available budget. With L2, the distribution of the goods sold value is much more widely spread across the tests. Whilst this graph does leave open the question of why MDA on L2 achieves such a varied number of goods sold across such a varied amount of spend it does show us that for the upper bound of results, MDA only increases the spend on goods as the number of goods increases (i.e. it never pays large sums of money for small numbers of goods).

By encouraging the traders to spending a low budgetary amount, we are encouraging strong behaviour from Buyers and lower prices for Sellers than would be achieved in a Combinatorial Auction. A distributed market where buyers are bidding for single items will only be able to force higher prices if demand for goods outstrips supply. If it did and prices rose then we would expect to see behaviour similar to the CASS results shown in Figure 6.5 with higher unit prices and a concentrated total spend.

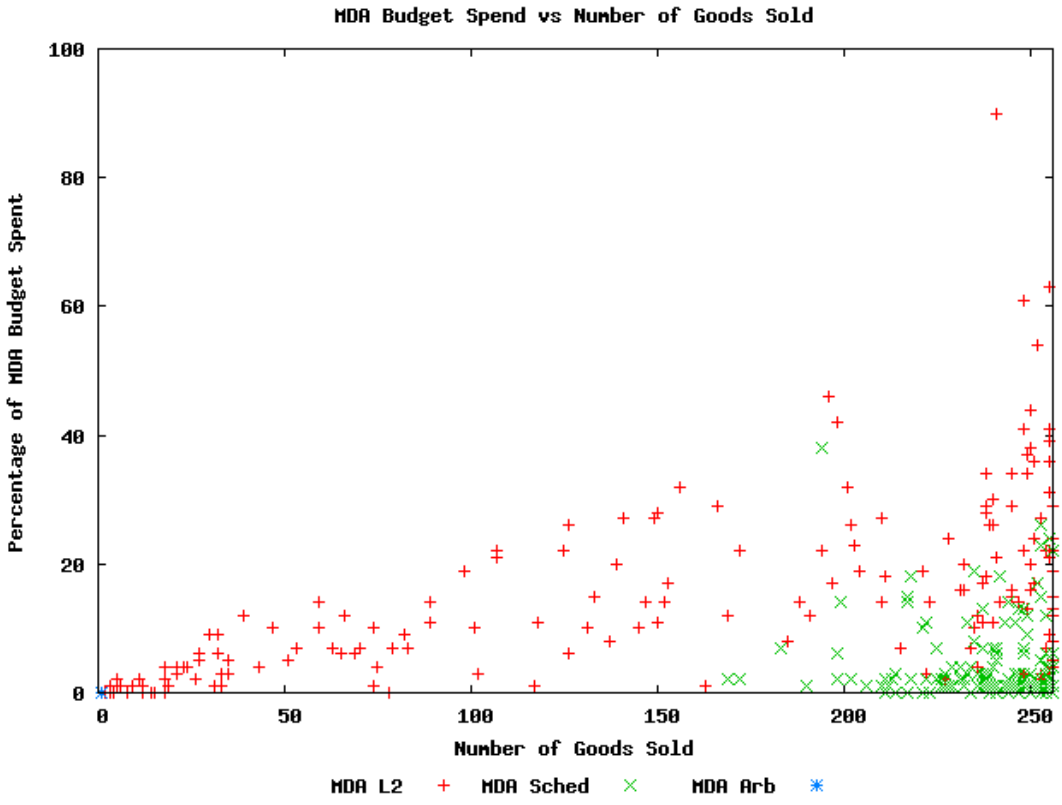

Figure 6.7: MDA Budget Spent vs Number of Goods Sold

6.5 Efficiency

This leads us to the question of market efficiency. In financial markets a market is considered to be efficient if it channels funds to those firms and organisations with the most promising investment opportunities, operating costs are as low as possible and the price represents all new and relevant information. In this context the focus is on the price of trades and specifically measures market efficiency as being the sum of total value of trades undertaken. In the CASS environment many of the bids are not met because they do not represent a Pareto optimal solution. However, that also means that there is value available to sellers which is not "unlocked" and potentially it would be possible to generate more value for sellers by complementing the complete bundles sold as part of a Pareto optimal outcome with the sale of goods from partial bundles. Considering Figures 6.8 and 6.9 we can see for both distributions, Scheduling and L2, both MDA and CASS have nearly equal efficiency for very hard problems, with a direct convergence in L2 (when sorted by Efficiency, we see that both datasets converge for instance 500). However, in L2, MDA is shown to be much more efficient for a larger proportion of the test data than CASS and it is important that we also consider run time,

because whilst the efficiency does converge for both distributions, the MDA algorithm achieves those results in much faster time, as we discussed in Figure 6.3.

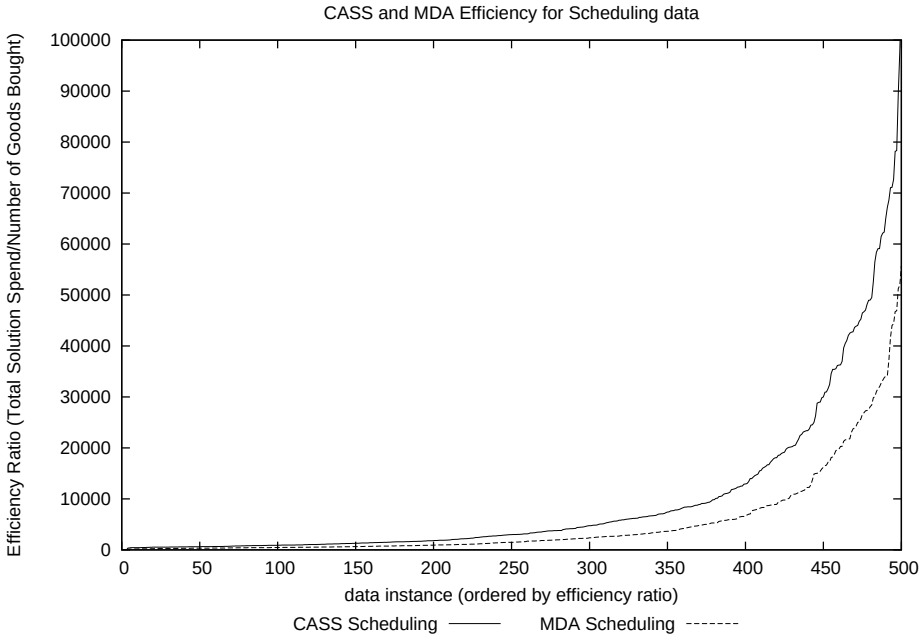

Figure 6.8: CASS and MDA Efficiency for Scheduling data

We have defined efficiency as the total sum of monies spent on goods acquired, divided by the number of goods acquired. Therefore, if the efficiency value is low, less money is spent per good (and ultimately more money is available to purchase further goods). When comparing CASS and MDA for the scheduling distribution (figure 6.8), we see that overall, MDA is more efficient, but as efficiency becomes less (cost per good increases) the results for CASS and MDA converge, showing that, for the worst case scenario MDA does not spend more per good than CASS.

The result for L2 (Figure 6.9) is more striking, with CASS always having a fairly high cost per good, but little variance in the average cost per good. The MDA solution has fewer results, because there are some problems for which MDA has no solution defined (as can be seen on the 0-50 results in Figure 6.9), but for many solutions, the cost per good, or efficiency, is much less. The average cost per good grows steadily, until MDA and CASS solutions converge for the final dataset, showing again that MDA is certainly not less, and in many cases more efficient.

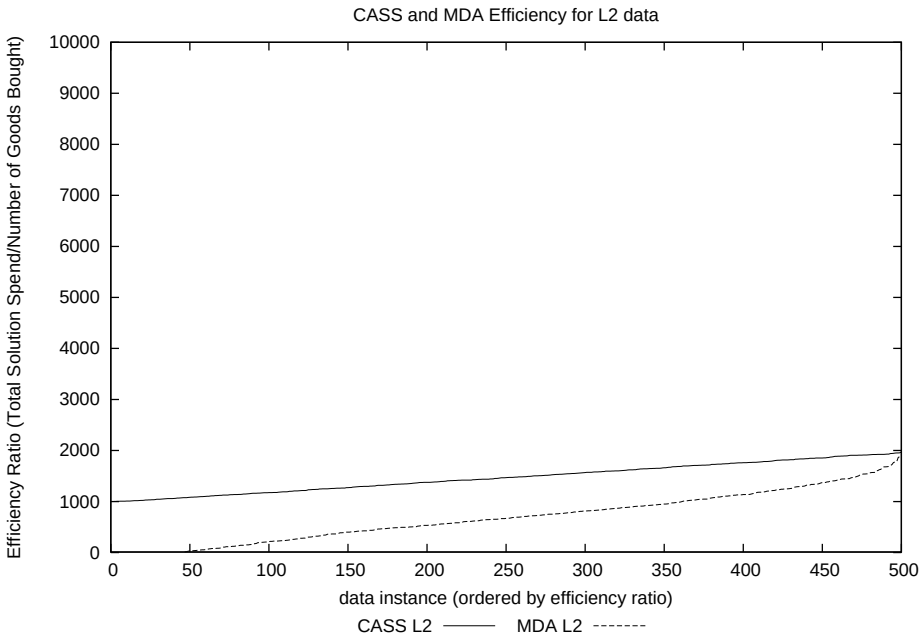

Figure 6.9: CASS and MDA Efficiency for L2 data

The efficiency figures are reinforced by a second set of results, those of the number of goods sold. In Figure 6.5, our graph of the total spend vs number of goods sold; we see that MDA is able to spend a significant fraction less than CASS. Interestingly, for the Scheduling distribution, all of the solutions solved by MDA have a fairly high number of goods sold (>200), whilst those solutions determined by CASS have a much wider spread from 10-250. However, the L2 distribution reverses this trend, with MDA results being spread widely, and CASS results being tightly clustered. Our runtime results give a hint as to why this behaviour exists, with our Scheduling problems being completed quickly by MDA, and L2 taking several orders of magnitude (as shown in Figure 6.3) longer.

This suggests that when a problem can be solved quickly by MDA, it may not be done so for significant extra savings on costs per good, but it will be solved with greater efficiency than CASS achieves. On problems such as those in L2 which a solver like CASS has traditionally found easy, MDA achieves better cost efficiencies but does take longer. With CASS the spend is higher and the time is shorter.

6.6 Completion

We define "Completion" as the percentage of the goods in a specific bundle purchased as part of the overall solution. Later, we can look at aggregate completion values, where we statistically analyse all of the completion values per test instance and per bundle.

To consider completion, we have looked at how many goods are traded, how many bundles are traded and which goods form those bundles. Each of 500 datafiles for the two distributions of CATS problem contains approximately 256 goods and 1000 bids for those goods. We have computed these problems using both available algorithms, CASS and MDA, which produces solutions, in the form of winning bids and a list of goods that were successfully traded to make up the bids for bundles of goods.

CASS always solves problems by ensuring it can allocate 100% of the goods required for those bundles that it wishes to trade, whereas MDA operates sequentially, with many agents buying goods and therefore partially completing the bundles. Some bundles are completed to 100%, but on average the percentage of complete bundles is much lower. We can observe that for the scheduling distribution, it is typically lower than for the L2 distribution.

The second important factor is the extent to which there is overlap between the goods that form the MDA and CASS solutions. This balance, of which goods, how many, for how much (financial result) and in what time, are the factors that combine to complete our understanding of the problem space.

6.6.1 Number of Bundles

We define the concept of a "touched bundle" which is a bundle that has some goods (>1) completed by either the MDA or CASS solver algorithms. We can therefore look at how many touched bundles there are per solution, what the percentage completion of those touched bundles is, giving us a representation of how many goods are sold and how many bundles are completed. Both MDA and CASS frequently complete different bundles and with CASS the bundles touched are always 100% completed.

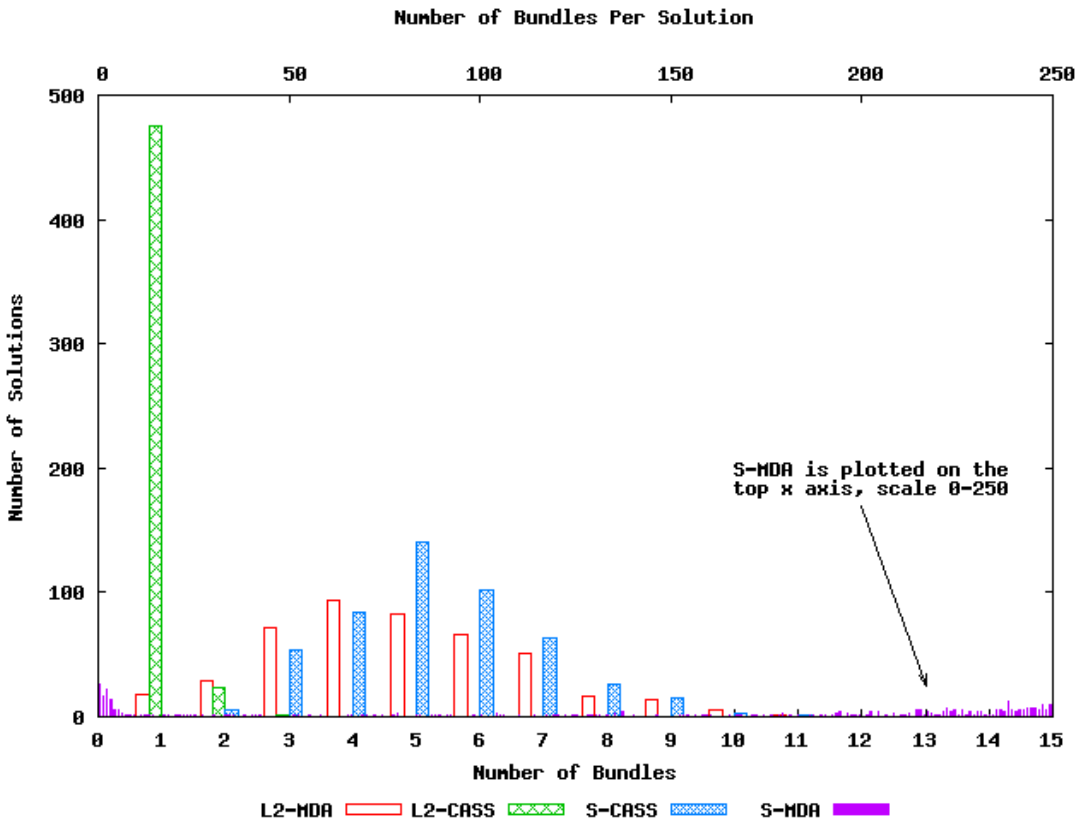

Figure 6.10: Number of Touched Bundles per Solution

Firstly, we wanted to look at the makeup of bundle completion for each of our 500 test instances, shown in Figure 6.10. For L2, we see that the majority of test instances have only one bundle (475/500) in the CASS generated solutions, whilst the CASS Scheduling solutions have a wider spread of completed bundles. For MDA, we find that there is a wider spread of number of bundles per solution and even more so for Scheduling, which has a very wide spread of bundles per solution. There are a maximum possible 1000 bundles in every CATS data file (for 256 goods) and so it would seem that if the algorithm is touching 250 bundles then the number of goods and the associated bundle completion ratio will be quite low. This appears to be consistent with Arrows Impossibility Theorem [AH71] as introduced in Section 2.2.

Whilst the bidders who seek those bundles find great satisfaction in that result, it is not representative of the satisfaction levels from the whole system and additionally, it is possible for MDA to trade a greater number of goods, across a wider range of bundles. In order to draw a comparison therefore, we need to compare the two systems in respect of the combined selection of bundles touched in their market processes.

6.6.2 Individual Bundle Completion

We have looked at how many bundles were touched for each solution which gives us some sense of the results spread, but we must look deeper and consider how the individual bundles are made up. Firstly, we consider the individual bundle completion ratio. We have computed the percentage completion rate for each individual bundle. For CASS, this is always 100%, as the CASS solver only produces solutions with complete bundles. MDA gives us information on actual goods traded and which bundle they were part of and by comparing those with the CATS datafile, we can count the percentage of goods in the bundle purchased. All goods are considered equal.

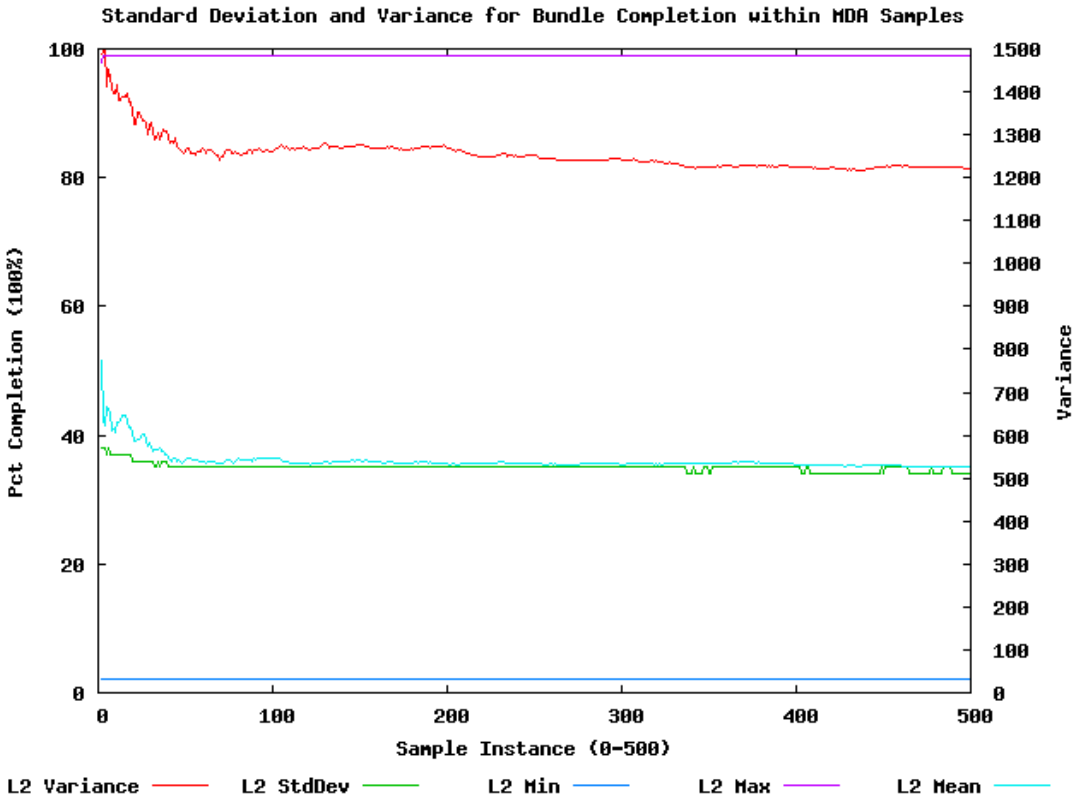

Figure 6.11: Standard Deviation and Variance for L2

In Figure 6.11 for the L2 data and Figure 6.12 for Scheduling we have derived the minimum, maximum, mean, standard deviation and variance for the set of completed bundles for each data file in the two distributions. For both sets of the data the first fifth of data has a much higher degree of variance and a much less consistent mean and standard deviation, an artifact, we believe, of the distribution used by CATS to generate the test data (each sample instance is computed through a separate run of the algorithm, so this is not a "warm up" phenomenon).

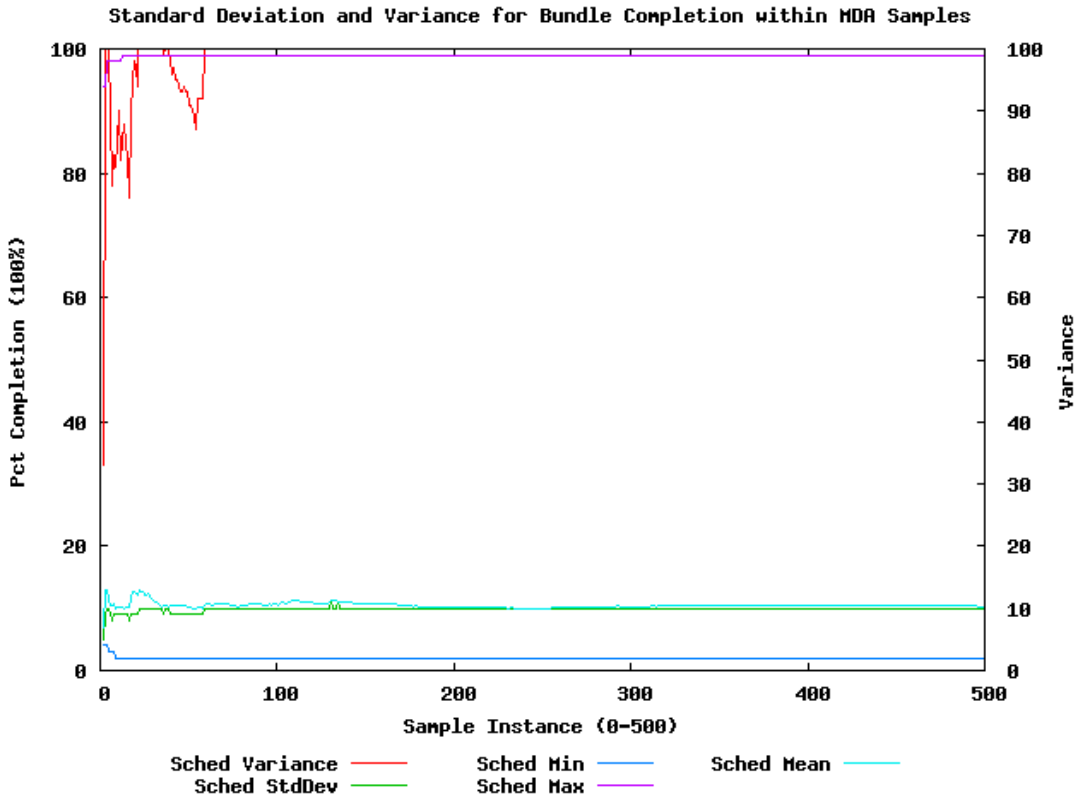

Figure 6.12: Standard Deviation and Variance for Scheduling

Looking at the figures, we see that there is a very high maximum and very low minimum, demonstrating that for all problems there are some bundles with near 100% completion and some with 0%. On average (mean) we see that L2 has a higher completion ratio than Scheduling data, however we also observe that for both we have high variance and that our standard deviation value is very close to the mean value. This shows a very wide spread of results and indeed, that none of the actual bundles complete with the mean percentage. This is information is replicated in Figures 6.13 and 6.15 which show an almost wall-like effect illustrating a very wide, but consistent spread of bundle completion. Indeed, it seems that looking at average data can be mis-leading. For example, we know that CATS source data is random, so it would seems unlikely that the percentage bundle completion should be so similar. However, we notice from the actual data (Figures 6.13 and 6.15) that there is simply a very wide spread of results, which derives from the randomness of the CATS data.

6.6.3 Which Bundles are traded?

Looking at figures in aggregate is not actually that useful then, as it only serves to show us that all datasets are consistently random! We need to consider the composition of the actual raw data. For our L2 data, Figure 6.14 introduces completion into the mix and shows us which of the touched bundles are included in the solution for each test instance. Each point on the graph represents a specific bundle of goods (with the Z axis representing the 1000 possible bundles). This is therefore a plot of the raw data behind Figure 6.10 but it now includes the percentage of the goods that bundle completed.

Figure 6.13 allows us to compare CASS and MDA results, with the CASS results forming a flat plane across the 100% completion level. For CASS we see a wide distribution of which bundles are used in solutions, but note that the density of bundles (represented by points on the graph) is reasonably sparse which correlates to our run time (Figure 6.3)—all the solutions are solved quickly and with a small number of bundles, indicating a quick, efficient search.

Figure 6.14 shows us, for each of the 500 L2 test instances computed by the MDA algorithm, which bundles formed part of the solution (Z axis) and what percentage of the goods requested in those bundles were successfully bought. The graph shows a small number of bundles achieving 100% completion and a much narrower spread of bundles per test instance.

For our Scheduling data Figure 6.15 again shows a similar wide spread of goods per bundle, but in line with the difference in the distributions structure, we see that MDA is able to increase the number of bundles utilised for some solutions and that CASS too has a similar wide distribution of goods used across all of the test instances and an increased density over L2. There is a much wider overlap of bundles in the solution between MDA and CASS, which helps alleviate one of our concerns, which was that in the L2 data, MDA is only handling the first 12 bundles for each solution, is potentially an artifact of an algorithm deficiency. However, as the same algorithm with a different data set behaves differently, we conclude that the limited range of L2 bundles used by MDA in its solutions is an artifact of the data.

We need to recall our runtime figures to understand this behaviour and introduce a Figure, 6.16 which compares the number of bundles touched in a solution, with the runtime of the experiment. We can contrast this “number of bundles” figure, with a

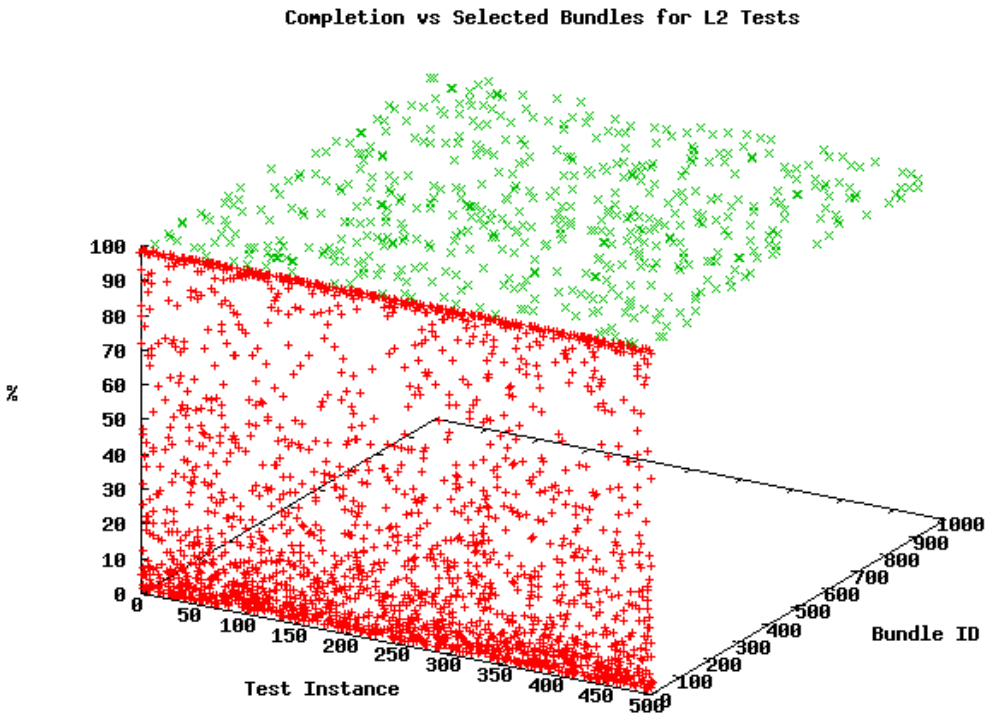

Figure 6.13: Completion vs Selected Bundles for L2 Tests

reprint (shown to the right) of Figure 6.4 showing "number of goods". Returning to the question, why do we see a greater overlap in which bundles are "touched" by the two trading processes for the Scheduling distribution but such a small cross over for L2? We propose that this is related to the extent to which both algorithms spend time examining all the bids—in the CASS model, this means the search space is relatively small. We can see, in terms of run time that L2 solutions for CASS take a small and concentrated amount of time, whilst CASS runs on the Scheduling data has a very wide distribution of runtimes, with many searches having to stop at our wall time. This wider spread of both run times, and the associated bundles touched and goods sold matches the huge amount of variance we get with other CASS results (such as financial) and our own MDA results. MDA run times are of course shorter, being related to the amount of time it takes for 500 rounds of the market, but we see a wide distribution again of the number of bundles touched for Scheduling, but not for L2. These graphs also show that CASS-L2 completed all runs within a very short time scale, but found solutions using a low number of bundles, illustrated with a low density of points in Figure 6.13, whilst CASS-Scheduling is again, higher. Finally, recall from figure 6.2 that deriving solutions for the problems takes longer in Scheduling than it does in L2. The greater

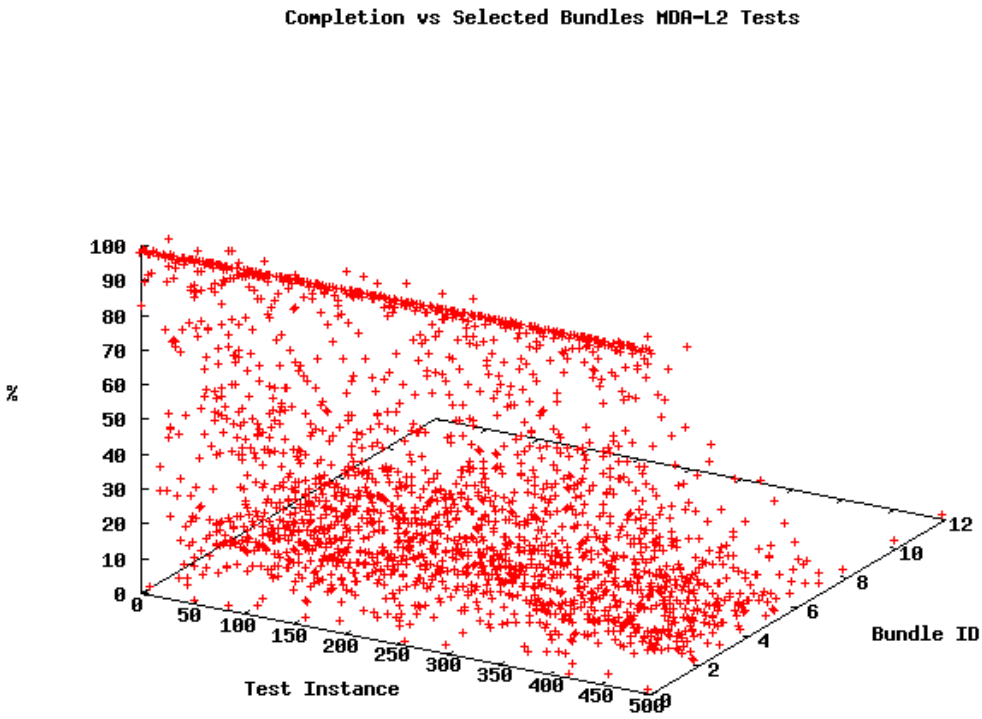

Figure 6.14: Completion vs Selected Bundles for MDA-L2 Tests

spread of solutions, for example, here make explicit reference to features of the plot illustrated in Figure 6.15 for scheduling over L2 supports that theory and the higher percentage completion suggests that MDA works well as a solver on more complex problems because the overhead of the market process is outweighed by the complexity of the problem.

It is argued that CASS produces optimum value and 100 percent completion, so therefore we do not need to consider the other goods or bundles left over, making the concept of touched bundles meaningless. However, optimum seller revenue and high completion rates for some bundles is not necessarily the best, or most suitable outcome for a particular problem space and we can see that the other elements of our comparison, such as speed, satisfaction and efficiency come into play depending on the properties of the problem space.

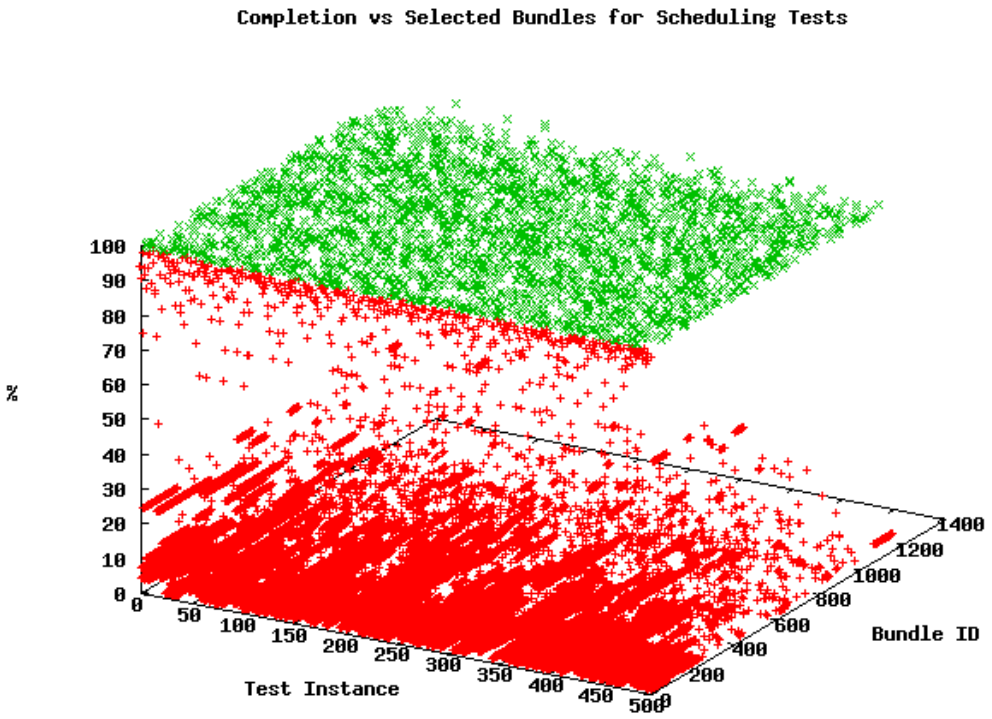

Figure 6.15: Completion vs Selected Bundles for Scheduling Tests

6.7 Satisfaction

We have examined, in some depth a number of factors: (i) Hardness, (ii) Time, (iii) Financial, (iv) Efficiency and (v) Completion . Through looking at all of these elements individually we have been able to draw correlations in data and understand the performance characteristics of the two algorithms, CASS and MDA and the two datasets, L2 and Scheduling. However, whilst this analysis gives us a menu of criteria which is helpful in making future choices over which algorithm to use for solving a particular problem, it does not universally answer "which is best"?

In order to complete the statistical comparison it would be helpful to have a single, combined metric by which we can measure the relative performance of the different approaches. We suggest that parameter is "satisfaction" and consider it to be the multiple of fiscal spend and the completion ratio for the bundle. The rationale can be illustrated as follows:

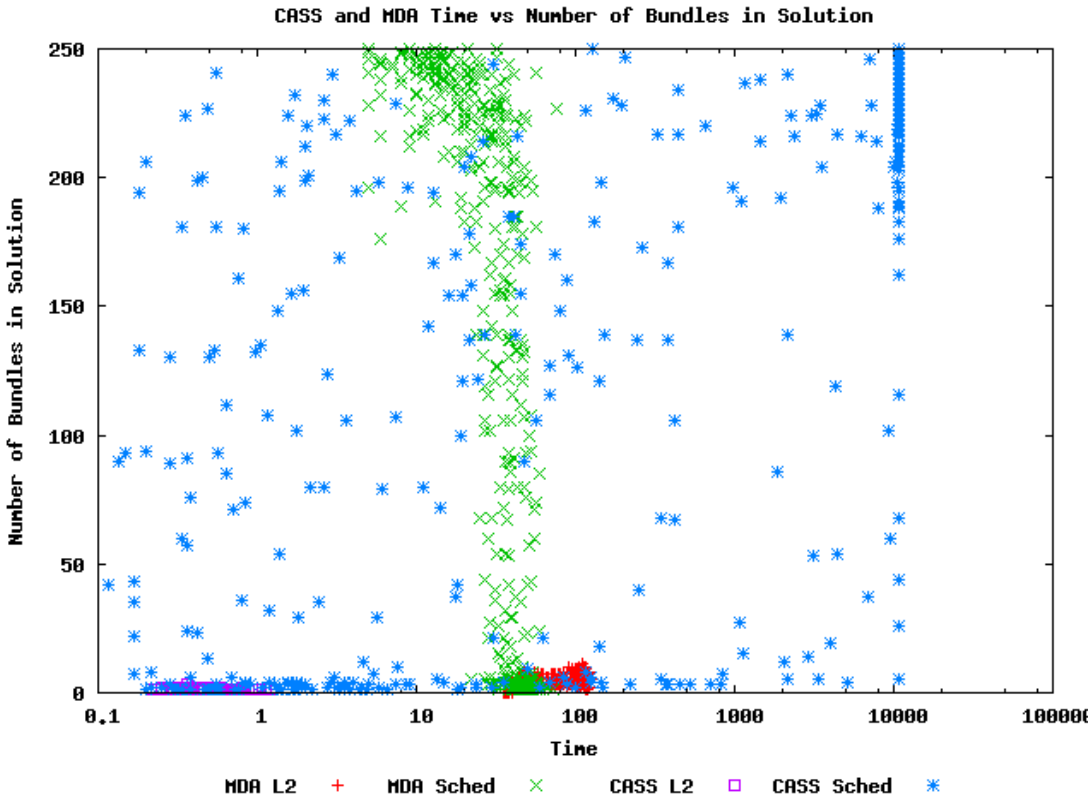

Figure 6.16: CASS and MDA Time vs Number of Bundles in Solution

- If I complete my bundle, I am happy. If I purchase it cheaply, I will be happier than if it has a high cost. If I get 100 percent completion for a high underspend value, then I will have a high satisfaction rating.
- If I get 50 percent of the goods desired for lots of money, I will have a very low underspend value which will be divided by 50 percent, giving me a very low satisfaction ratio.
- If I get 50 percent of goods desired for little money, I will be reasonably satisfied, so my high underspend value will be divided by two thus taking the downside of an incomplete bundle into account.

In Figure 6.18 we show our satisfaction ratings against the spend on all bundles in the final solution, for each test instance. The figure shows a number of interesting elements, firstly, that for the CASS results, because we are essentially multiplying the percentage of goods used in the sold bundles by 100%, the satisfaction increases linearly as spend increases against the offset provided by the goods used value. We see again, the tight-bounding of the L2 results and the wider spread of the Scheduling

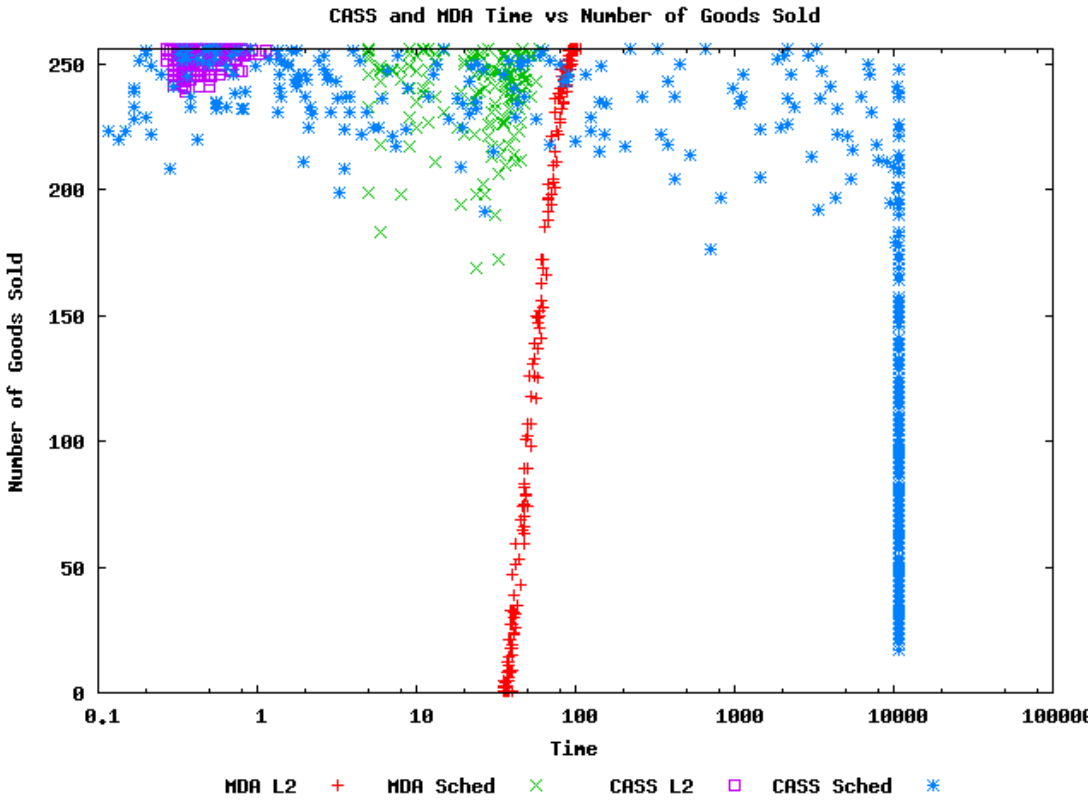

Figure 6.17: CASS and MDA Time vs Number of Goods Sold

results.

For MDA satisfaction, the results are more interesting. We know that the percentage of goods purchased (out of all requested purchases) is much more variable for MDA solutions, but because, overall, the number of goods sold is greater than for CASS solutions and the financial spend on goods is lower for the distributed market approach, satisfaction ratings start higher and are not related directly to spend.

6.8 Conclusions

All measures of this sort are open to interpretation and arguably, the satisfaction graph is an alternative representation of Figure 6.5, which shows CASS and MDA Actual Spend vs Number of Goods Sold. However, this data, when taken in context with the other aspects and assessments we have presented serve to illustrate that both distributed markets and combinatorial auctions have valid uses and different properties which

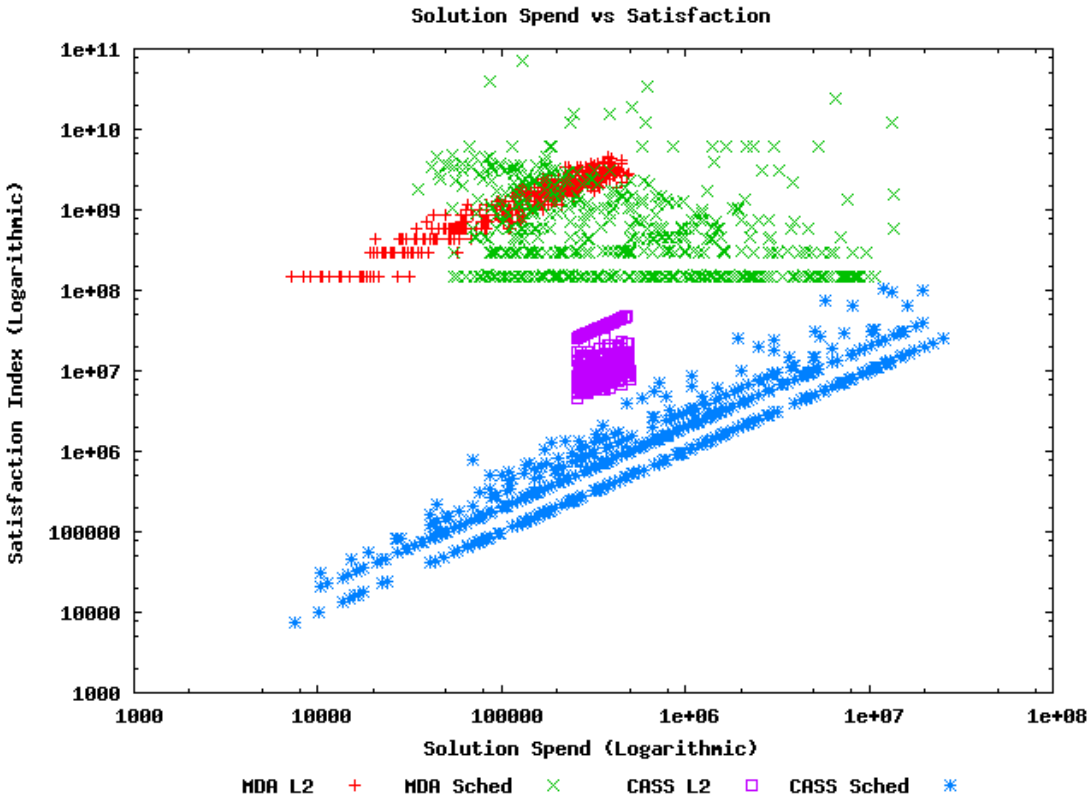

Figure 6.18: Budget Spent vs Satisfaction

make them suitable for different applications.

The question of "what makes a distribution hard?" has much literature devoted to it, and one of the obvious mechanisms, given that a combinatorial auction is an NP-Hard problem, is to consider the timing requirements for solving individual problems. However we discovered (Figure 6.3) that when comparing KLB's published CPLEX and CASS results CASS found the L2 distribution easier than CPLEX but the Scheduling distribution was considerably harder (with a much wider spread of runtime magnitudes). Our MDA results had more consistent run times, as the MDA algorithm ran for 500 iterations of the distributed market and so for MDA, and increase or decrease in time was only due to fluctuations in the volume of trades for each round and so the different distributions have a much smaller effect on timings than can be seen on CASS and CPLEX results. Returning to the question of hardness, it seems to us that time itself is also not a strong measure of difficulty and that improvements and differences in algorithms and test data distributions, even within two combinatorial auction solvers, can lead to differences in reporting performance.

Unfortunately, despite exhausting many thousands of hours of compute time, we were not able to run all the CASS experiments for our 1000 test results to completion. This introduced a wall effect which we could see clearly in our runtime comparisons. The effect of having a wall in the CASS algorithm is that it is unable to complete the searching of its result tree and may produce suboptimal answers. Looking at runtimes in isolation however, we concluded that MDA was more predictable.

The next test was that of the financial performance of the markets. Comparing MDA and CASS results, we saw that MDA spent less on goods (both overall and per good). This was markedly noticeable (Figure 6.6) for the Scheduling distribution, but was also prevalent in L2 and is evidence of the two strategies (markets maximise for buyers, auctions for sellers) working effectively. Finally, we noted that MDA's linear approach to good purchasing means that it will never pay a large sum of money for small numbers of goods—which can be important in situations where it is necessary to purchase bundles with very large numbers of goods.

One of the trade-offs presented by the two approaches is whether or not to maximise utilisation of goods, or revenue. We considered market efficiency, defined as the total value of trades taken. For both distributions, Scheduling and L2, both MDA and CASS have nearly equal efficiency for very hard problems, with a direct convergence in L2. However, in L2, MDA is shown to be much more efficient for a larger proportion of the test data than CASS and it is important that we also consider run time, because whilst the efficiency does converge for both distributions, the MDA algorithm achieves those results in much less time.

The notions of time, financial outcome and efficiency are all useful tests to help us understand how our markets are performing, but ultimately, if the resource allocations produced are of no use, the price and time taken does not matter. We therefore looked at the issue of bundle completion and how many goods are traded, how many bundles are traded and which goods form those bundles.

Looking at the spread of bundles per solution, we found that MDA traded a wider quantity of bundles for both the L2 and Scheduling datasets and to a lower level of bundle completion. We observed a completely random spread of bundle allocations in both CASS and MDA solutions, reflecting the random nature of the good allocations in the original CATS data. We also concluded that when dealing with a random set of preferences, against which there is no standard distribution, we would find high

statistical variances for aggregate measures such as the mean percentage of goods purchased to complete a bundle. We found it more appropriate to consider the raw data and look at which goods specifically were traded and for the Scheduling dataset we found there was a considerable overlap between bundle selection by CASS and MDA, whilst for L2, the MDA solutions all utilised the initial goods in the test data (elements 0-12) whilst CASS solutions were much more incomplete, again, the two mechanisms showing different approaches to managing data, with CASS failing to compute solutions with many goods whilst MDA attempts to complete a larger number of bundle requests, spreading goods resource thinly, but maintaining good utilisation.

These graphs also show that CASS-L2 completed all runs within a very short time scale, but found solutions using a low number of bundles, illustrated with a low density of points in Figure 6.13, whilst CASS-Scheduling is again, higher. Finally, recall from Figure 6.2 that deriving solutions for the problems takes longer in Scheduling than it does in L2. The greater spread of solutions illustrated in Figure 6.15 for scheduling over L2 supports that theory and the higher percentage completion suggests that MDA works well as a solver on more complex problems because the overhead of the market process is outweighed by the complexity of the problem.

Finally, we utilised an aggregating metric which attempts to capture a potential buyers satisfaction with a set of results. Through combining spend and completion measures we showed that because, overall, the number of goods sold under MDA is greater than for CASS solutions and the financial spend on goods is lower for the distributed market approach, satisfaction ratings start higher and are not related directly to spend, which is an important approach if, as a buyer, you are attempting to maximise your purchases.

Looking at our test data from the six dimensions we are conscious that, for any specific measurement it is difficult to identify the pros and cons of our two approaches—distributed markets and combinatorial auctions.

However taken together we contend that a distributed market provides an approach for solving more difficult problems, for example the job shop scheduling task simulated by our Scheduling distribution. They ensure that resource usage is maximised, with good efficiency, revenue and in a predicable time frame. Combinatorial solutions will always produce a Pareto efficient option and maximise revenue for the sellers of goods, but their run times vary unpredictably based on both the algorithm used and the complexity structure of the problem, the result being that resource utilisation is often sacrificed.

Chapter 7

Further Experiments

7.1 Introduction to Further Experiments

Kevin Leyton-Brown (KLB) saw that many researchers had proposed algorithms for determining the winners of a combinatorial auction, which itself gives rise to the problem of how to evaluate, compare and therefore improve the algorithms in an objective manner. It is this problem that he sought to solve in his PhD thesis [LB03] and he outlines the most recent version of the Combinatorial Auction Test Suite (CATS) in [CSS05, Ch.18]. We selected the L2 and Scheduling distributions from the available data in order to produce the comparisons and experimental results in this thesis, because they most closely represented the structure of the resource allocation problem genre that we were focused on providing solutions for.

CATS provides a number of legacy datasets, denoted by "L". These were introduced for comparability and are derived from work by Sandholm [San02], de Vries and Vohra [dVV03], Anderson [ATY00], Boutilier [BGS99] and Fujishima [FLBS99]. These legacy distributions had variances in the model used to calculate the number of goods (is it uniform across samples, a normal distribution, a decaying distribution); which goods are selected for the bundles (is it done randomly, or are all used); and how are the prices for each bundle in the distribution set (randomly, linearly or normally)? However, these distributions all suffer from weaknesses and particularly contain large numbers of non-dominated bids which makes them empirically easy to solve and a

poor choice for computational benchmarking [1].

Leyton-Brown looked to tackle the weaknesses in the legacy distributions and proposed model distributions which were derived using a range of different techniques in order to represent a number of sample problems:

Paths: Problems related to the purchase of a set of connecting points, such as logistics or sales people routing, network bandwidth allocation, or railway track scheduling.

Regions: A class of problems in which goods derive complementarity from their adjacency, such as sale of real estate, drilling rights in oil fields or radio/telecom spectrum auctions.

Scheduling: The scheduling distribution deals with the classic job-shop scheduling problems, whether in factories or grid computing environments each user has a job requiring some amount of machine time and a deadline by when it should be completed.

Arbitrary: A general model of arbitrary relationships where the complementarity is not as obvious as physical adjacency, but instead might be related to the production of a larger unit. Example problems would include the purchasing a collection of electrical components, a set of collectible antiques or the right to emit a quantity of industrial pollutants (carbon emissions trading).

Matching: There are a number of real world domains where complementarity between goods arises because of time related considerations and therefore they must be matched together. As an example, development of an effective mechanism to sell take off and landing slots in corresponding airports by combinatorial auction is currently an unsolved problem.

We have attempted to compute the results for CASS and MDA for the extra four data distributions discussed, but present them in summary, for discussion, rather than in any detail or in support of the main arguments of this thesis. We do that for the following reasons:

[1] We use L2 because it is the most widely examined legacy distribution and therefore provides a recognisable distribution for comparison

(i) The structure, make up and "shape" of the distributions of goods is one of the key factors that affects the complexity of computing the solution, for both the MDA and CASS algorithms and it is not possible to fully understand their results without in-depth reference to the explanation of each of the distributions given in [CSS05, Ch.18], and it is not practical to reproduce that here.

(ii) Computing the additional data required months of computation time (significantly longer than estimated) and due to various practical problems in regards access to facilities, the results arrived towards the end of the production of this thesis document.

(iii) Trying to compare and contrast the results from six data distributions and two algorithms presents significant data visualisation challenges that would require significant effort to resolve, for which time was not available.

(iv) In the context of this thesis, where our objective has been to determine the most appropriate method to be used in order to compute the best allocation of a generic set of resources and to develop an under-standing of how the different approaches available are related to one another, we feel that an in-depth analysis of the two datasets (L2 and Scheduling) is more than sufficient to explore the characteristics of MDA and CASS and that whilst these results confirm and support the discussion in Chapter 6 they do not enhance it.

Therefore, in the remainder of this Chapter, we present a range of information and supporting notes for future discussion.

7.2 Hardness

Hardness of the distributions is considered to be related to the runtime order of magnitude and KLB produced his version, shown as figure 6.1 (page 109) which we have emulated in Figure 7.1. This graph shows the runtime, by Log_{10} order of magnitude for all distributions computed and shows us that CASS typically finds the new distributions take an order of runtime longer to compute (frequently hitting our wall clock time) than they do with CPLEX. MDA still runs for it's standard amount of runs.

The new distributions can be considered "hard", particularly arbitrary, matching and

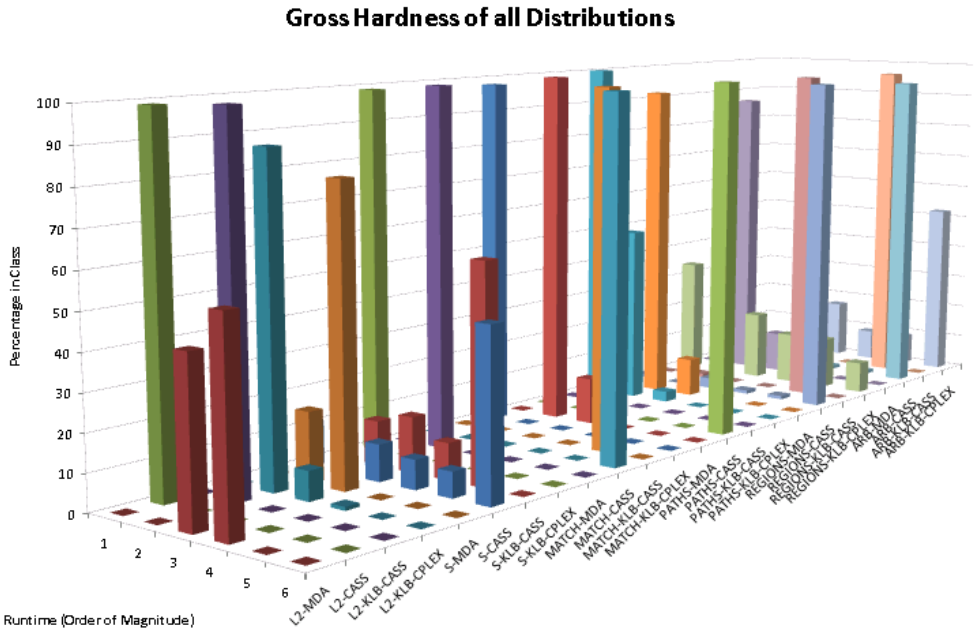

Figure 7.1: Gross Hardness Distributions

regions, as KLB's own results shown in Figure 7.2 show, with a large percentage of the solutions shown in the runtime class 5.

To repeat our previous analysis, we have reproduced KLB's published CPLEX solutions in Figure 7.3, which in comparison to the CASS results 7.2 reinforces the point that CPLEX and CASS have different performance characteristics (despite solving the same problem).

7.3 Number of Goods/Bundles

For the runtime vs number of goods or bundles sold graphs we have split the visualisation of the data and show CASS and MDA results separately. Additionally, when reviewing the graphs in this section, the concentration of data points is an indicator of the number of instances of a problem that were solved by the algorithm in the time allowed. As with our main results, we computed 500 test solutions for each data distribution.

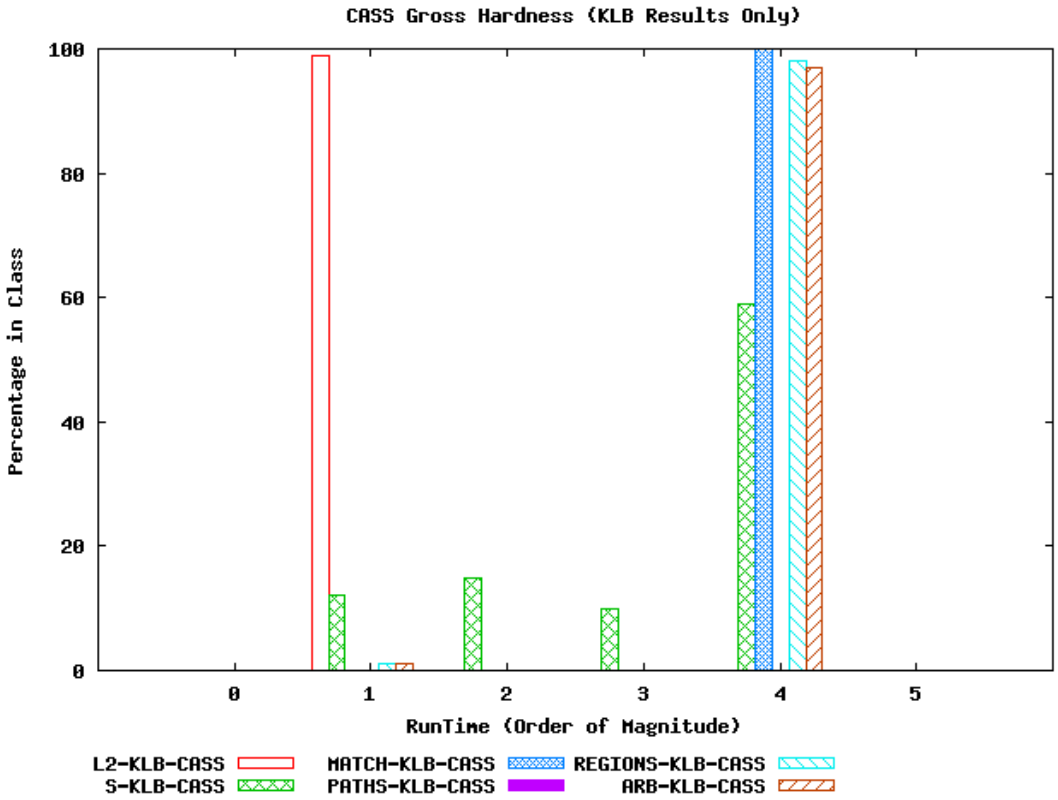

Figure 7.2: CASS Gross Hardness (KLB Results Only)

(i) The CASS graph in Figure 7.4 shows a thin spread of results for arbitrary and regions data, suggesting that for these there are few instances with good sold and that many of the computations reached the wall clock time without completing.

(ii) The Paths data on figure 7.4 is shown to have a very narrow time spread. When compared to figure 7.5 we see that whilst goods are traded, bundles are not completed in the final solution, suggesting that CASS finds the problems hard to compute.

(iii) The scheduling distribution has the widest distribution of solutions completed across the time window showing the random nature of the problem.

(iv) The matching distribution is only present at the top level of the "wall column" in both figure 7.4 and figure 7.5, suggesting that many goods and bundles are utilised in these problems, but at the expense of a lot of computing time.

The two MDA figures compare number of goods sold (figure 7.6) and the number of bundles in computed solutions (figure 7.7).

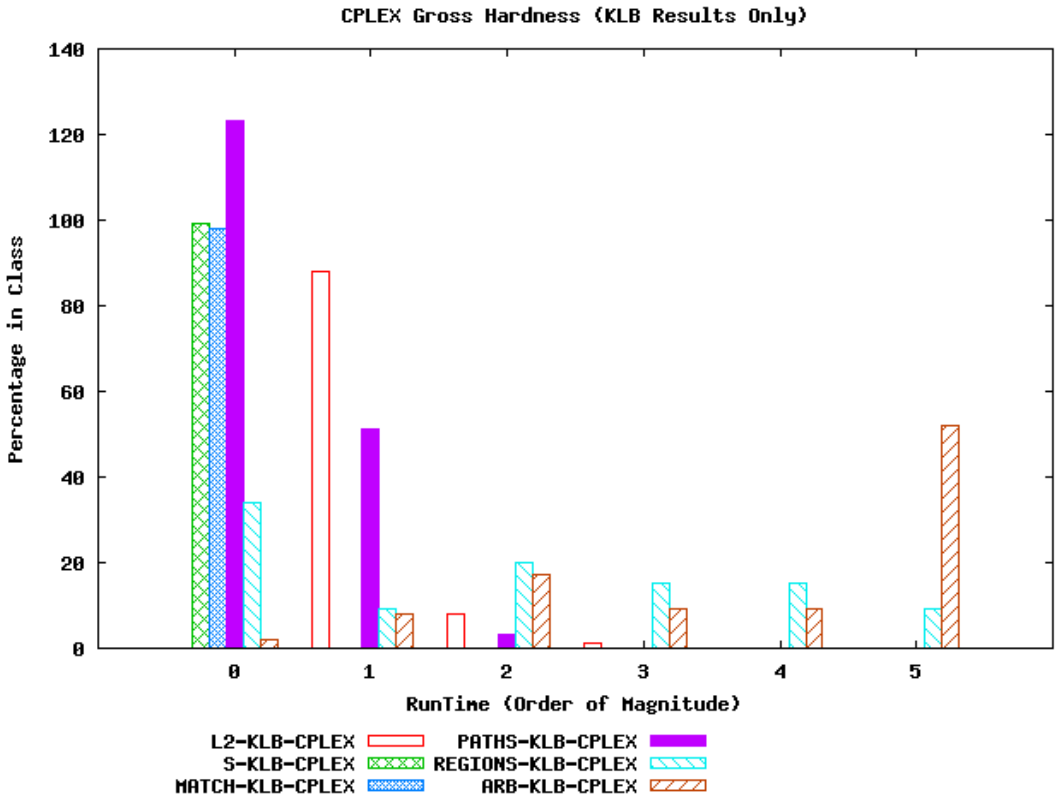

Figure 7.3: CPLEX Gross Hardness (KLB Results Only)

(i) L2 data shows a low number of bundles completed, a high number of goods sold, suggesting good completion ratios.

(ii) Scheduling data shows a wide spread of bundles but a narrow time window, suggesting all problems are of similar hardness to solve. The vertical spread is similar for MDA and CASS.

(iii) Regions data shows a sparse number of bundles and no goods sold by MDA, suggesting that the MDA algorithm is not suited to solving these problems in a short number of rounds.

(iv) Matching data shows a high number of goods sold and bundles computed in a short time frame, suggesting MDA is a good solution for solving Matching problems.

(v) Paths, as with Regions, we do not have Paths data for goods/bundles sold.

(vi) Arbitrary data shows that a small number of problems did complete, but that arbitrary problems take longer than the time available to compute, as can be seen for the CPLEX and CASS hardness in Figure 7.1

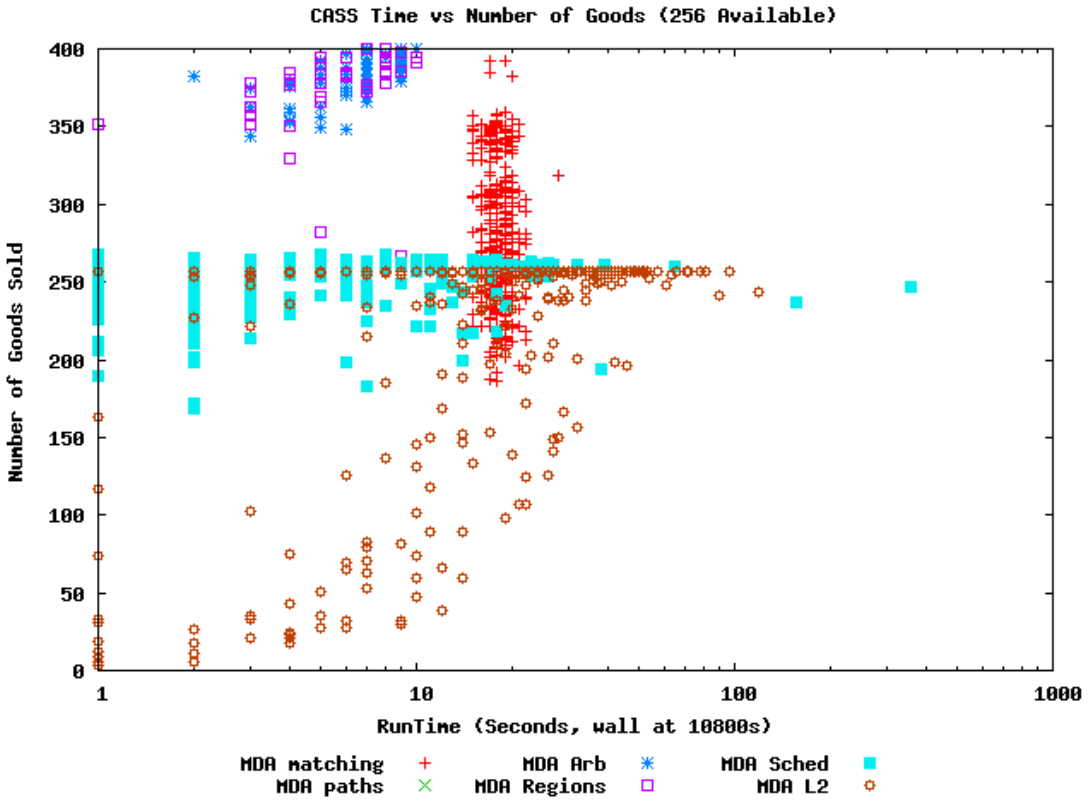

Figure 7.4: CASS Time vs Number of Goods Sold

7.4 Actual Spend

The actual spend graphs in figures 7.8 and 7.9 illustrate the amount of monies spent by the traders in buying the goods (for MDA) or the cost of the goods selected (for CASS) in order to compose their bundles.

(i) For the spend graphs, being in the top left hand corner would be the optimum position (high good utilisation, low spend)

(ii) The CASS graph (figure 7.8) shows that CASS is optimal fgor Paths and Arbitrary, whilst the MDA results in Figure 7.9 suggest it can solve Matching and Scheduling problems more easily. L2 is solved well on both, with marginal improvements in the number of goods sold (but not spend) in CASS.

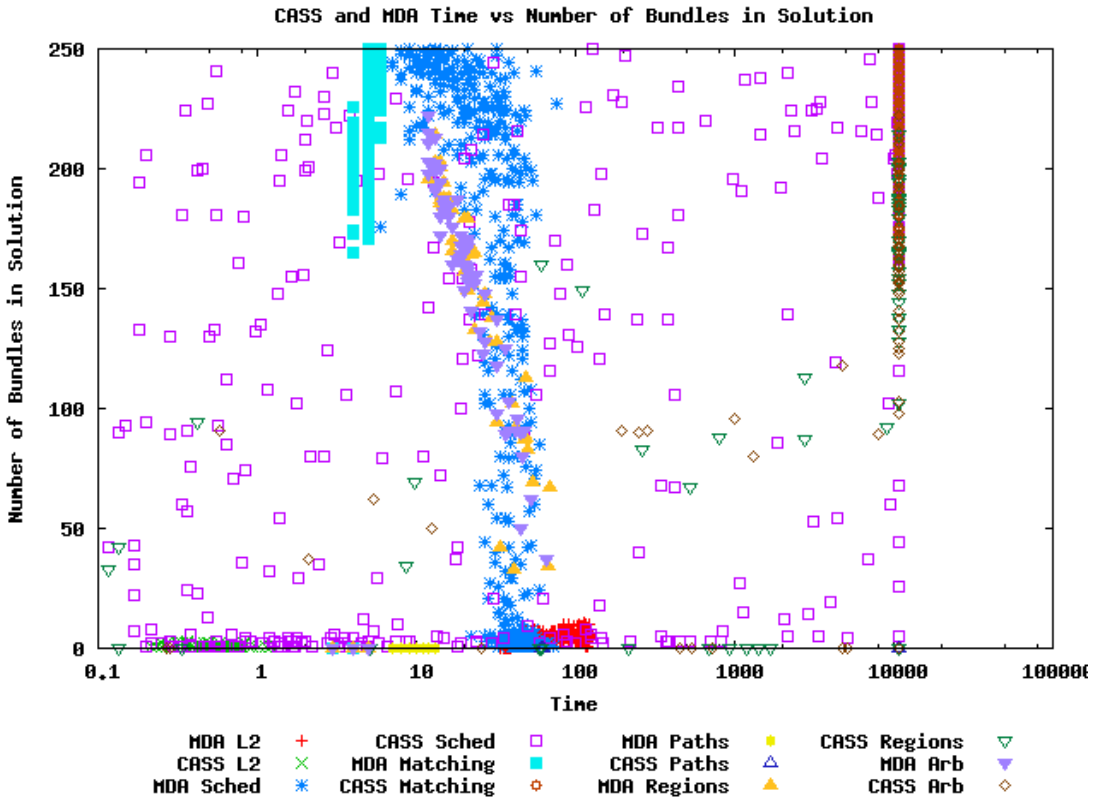

Figure 7.5: CASS Time vs Number of Bundles in Solution

7.5 Efficiency

The efficiency graphs use our calculated efficiency ratio, which is the solution spend divided by the number of goods bought as a comparison factor.

(i) MDA did not produce results for the number of goods bought for Paths, Arbitrary or Regions, and therefore does not have MDA efficiency values.

(ii) For Paths data, we see that efficiency increases in line with the number of goods sold for CASS.

(iii) The CASS arbitrary data indicates that there is high efficiency, but for a small number of problems—arbitrary problems are hard to solve!

(iv) Scheduling has a wide spread of efficiency values, but MDA appears to sell more goods for similar efficiency than CASS.

(v) L2 has poor efficiency on CASS and is more efficient under MDA.

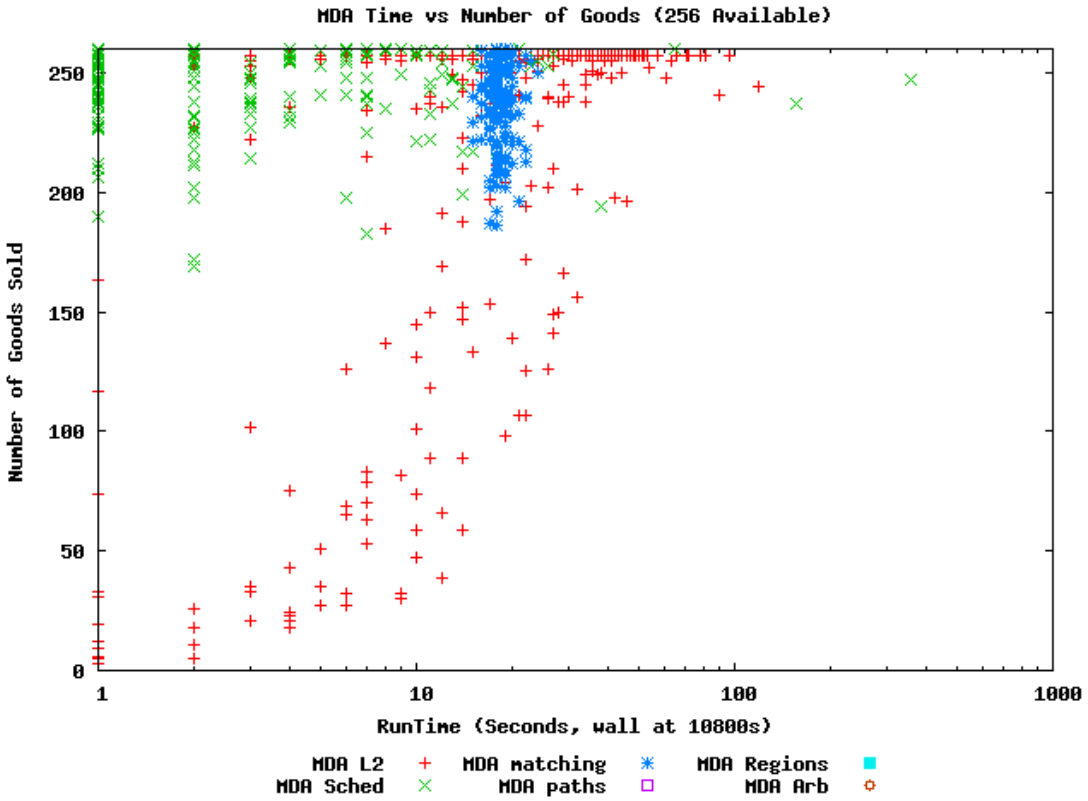

Figure 7.6: MDA Time vs Number of Goods Sold

(vi) The Matching distribution problems are solved with a high degree of good utilisation by MDA, but there are minimal problems solved under CASS.

(vii) Regions data is not visible on either graph with the underlying data suggesting that no regions goods were solved.

7.6 Satisfaction

Figure 7.12 and 7.13 show our satisfaction metric and solution spend.

(i) For all the CASS data sets, we see that satisfaction and spend rise linearly together, against both logarithmic axis. This tells us that for all CASS based algorithms high levels of satisfaction can only be obtained with similarly high spends.

(ii) CASS finds the Paths and Scheduling distributions "easiest" to produce good solutions for whilst regions and arbitrary are the most difficult.

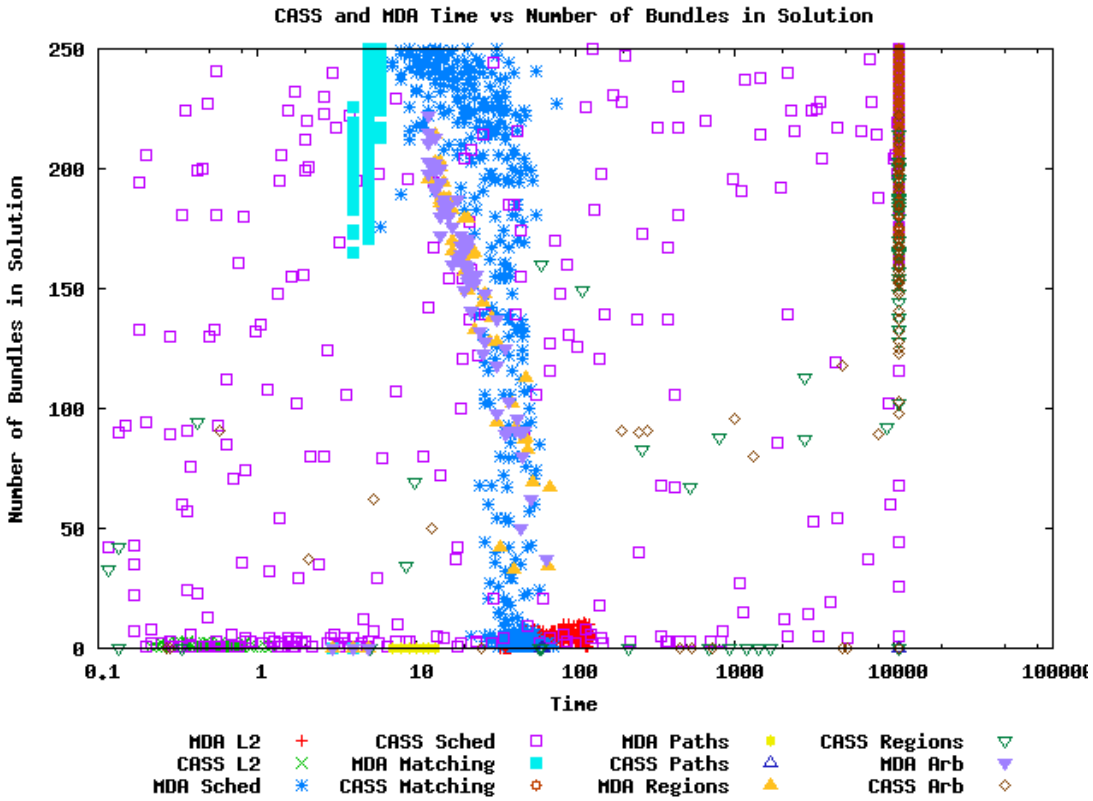

Figure 7.7: MDA Time vs Number of Bundles in Solution

(iii) With the exception of L2, all MDA results have much tighter clustering and a number of horizontal plateaus (on a logarithmic scale) suggesting that MDA scales linearly with the complexity of the problem space being solved.

(iv) For the L2 distribution, MDA derives higher satisfaction for lower spend.

(v) For Matching distribution, the problems are clustered into two regions, but with linear increase in spend for MDA, rather than logarithmic for CASS.

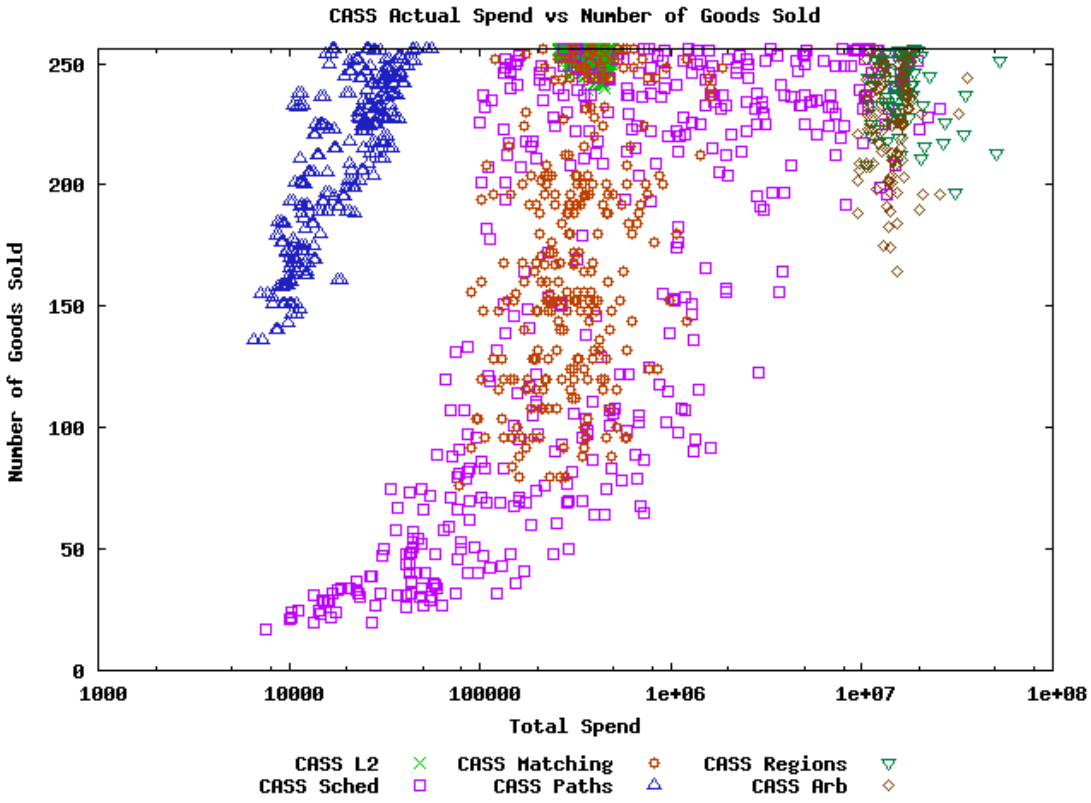

Figure 7.8: CASS Actual Spend vs Number of Goods Sold

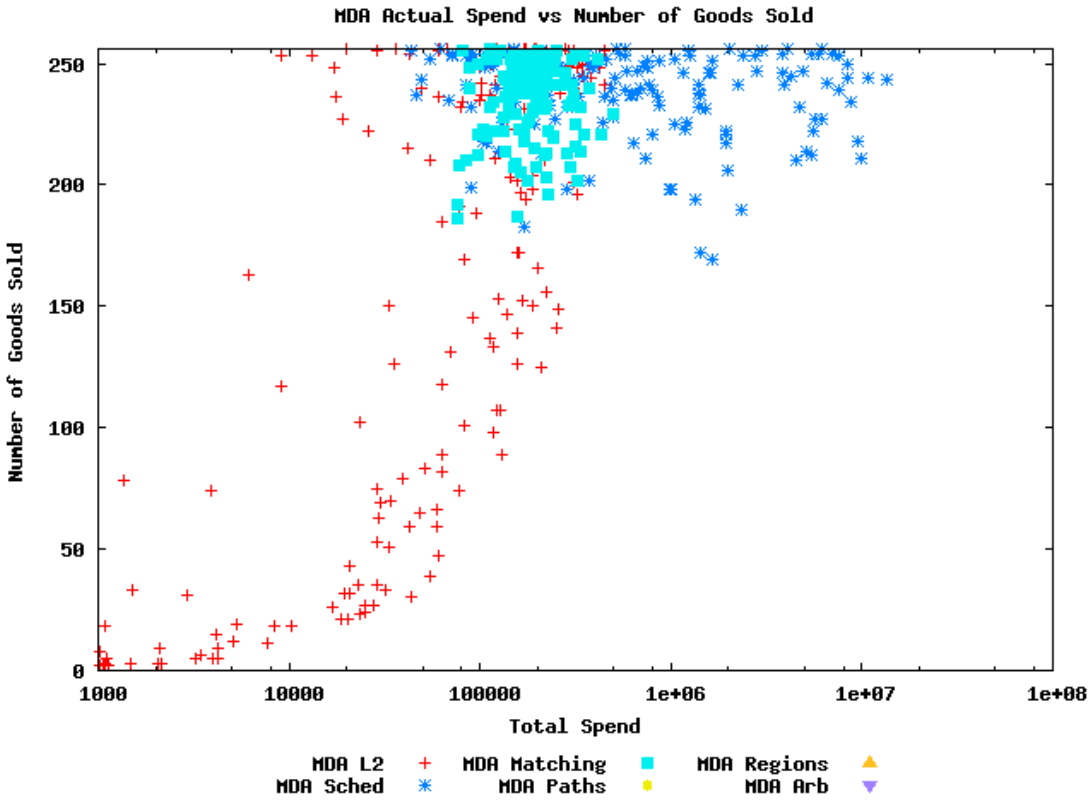

Figure 7.9: MDA Actual Spend vs Number of Goods Sold

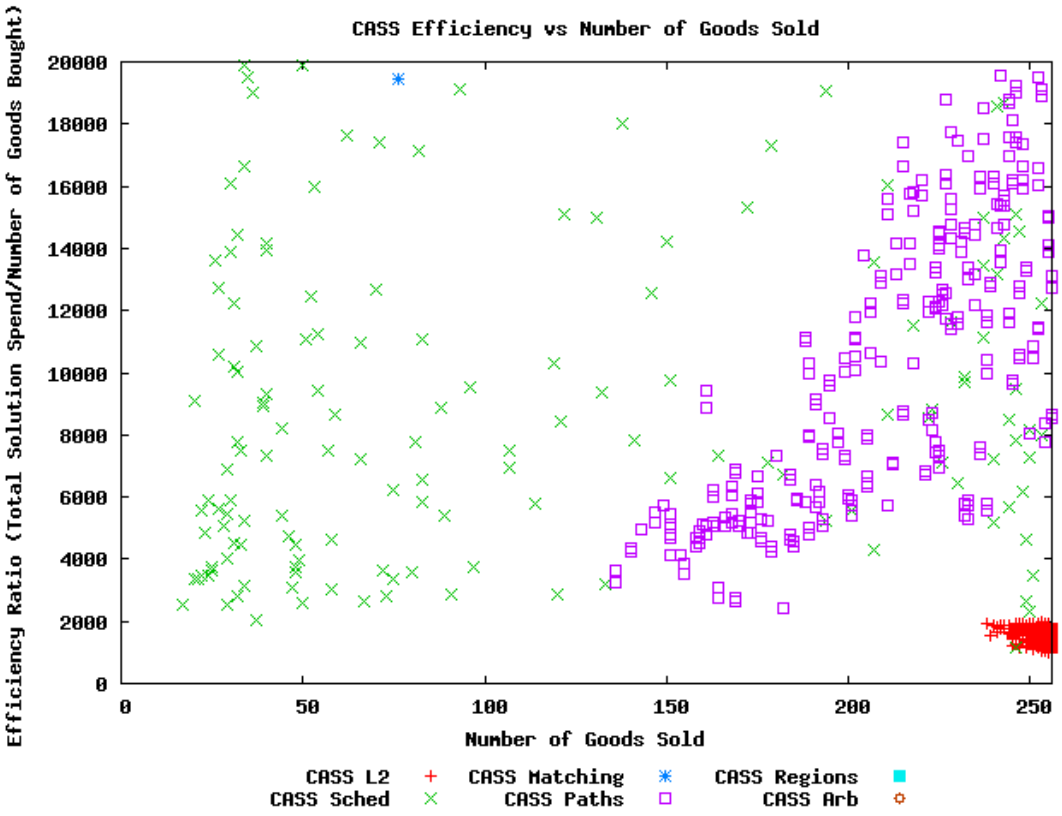

Figure 7.10: CASS Efficiency vs Number of Goods Sold

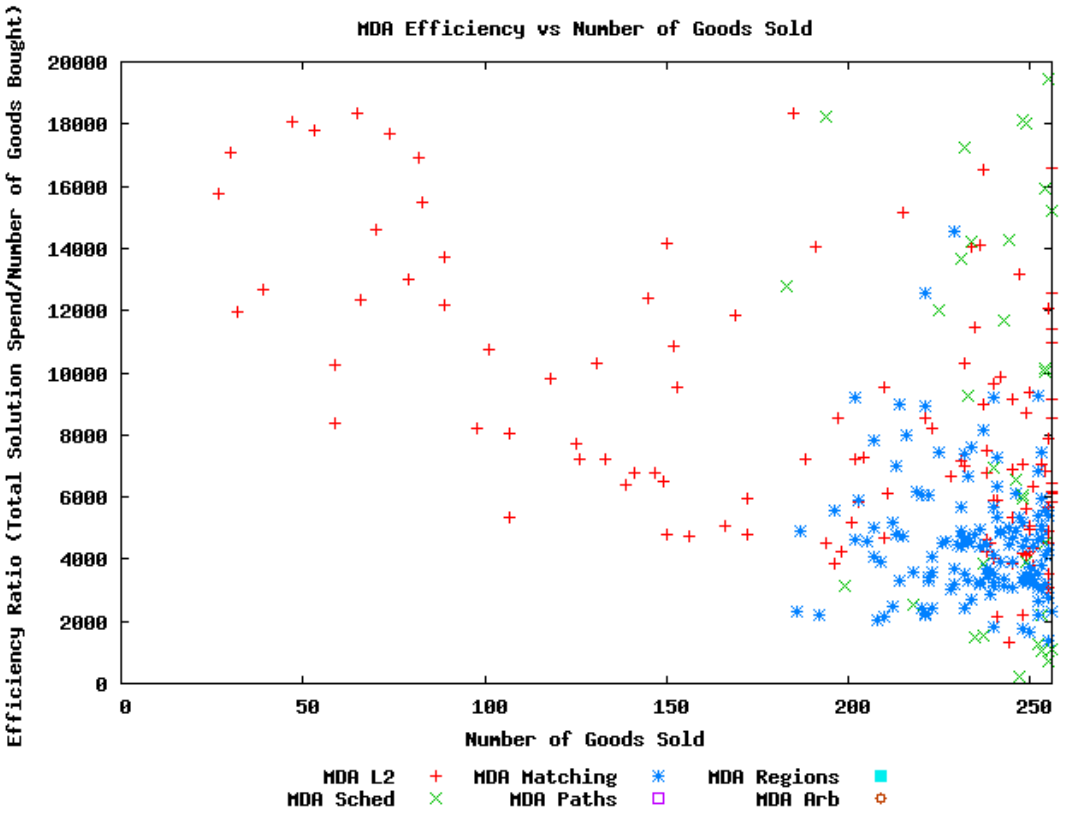

Figure 7.11: MDA Efficiency vs Number of Goods Sold

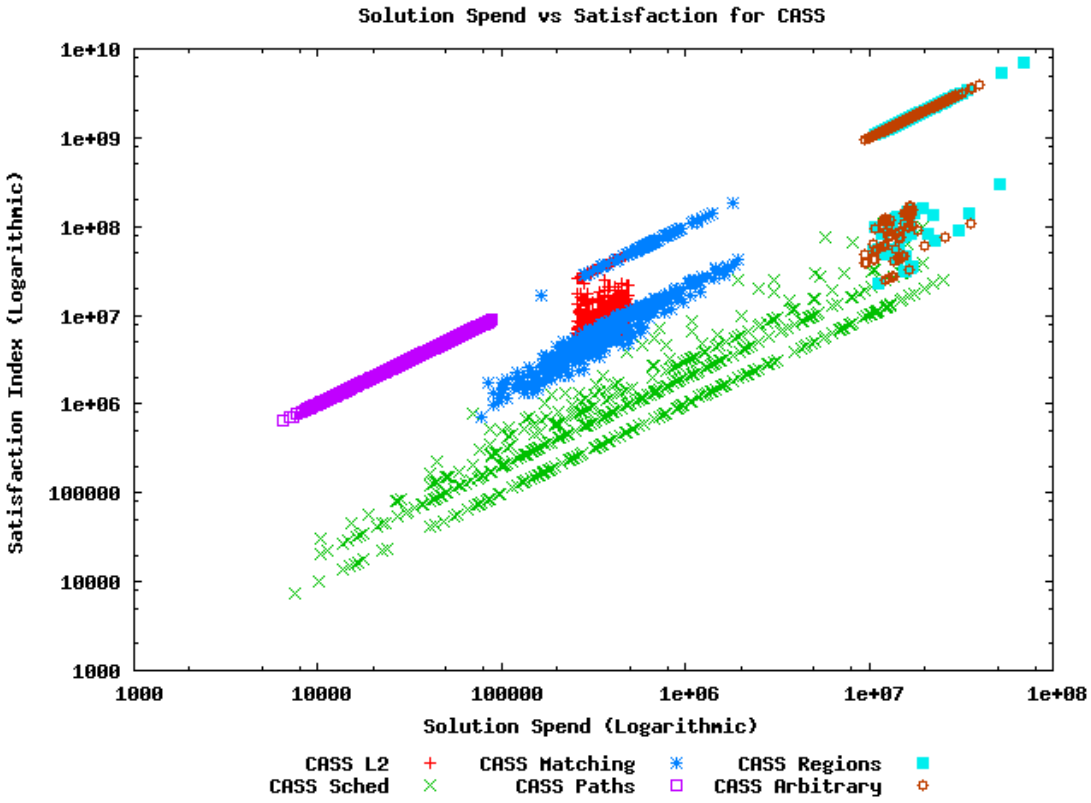

Figure 7.12: CASS Solution Spent vs Satisfaction

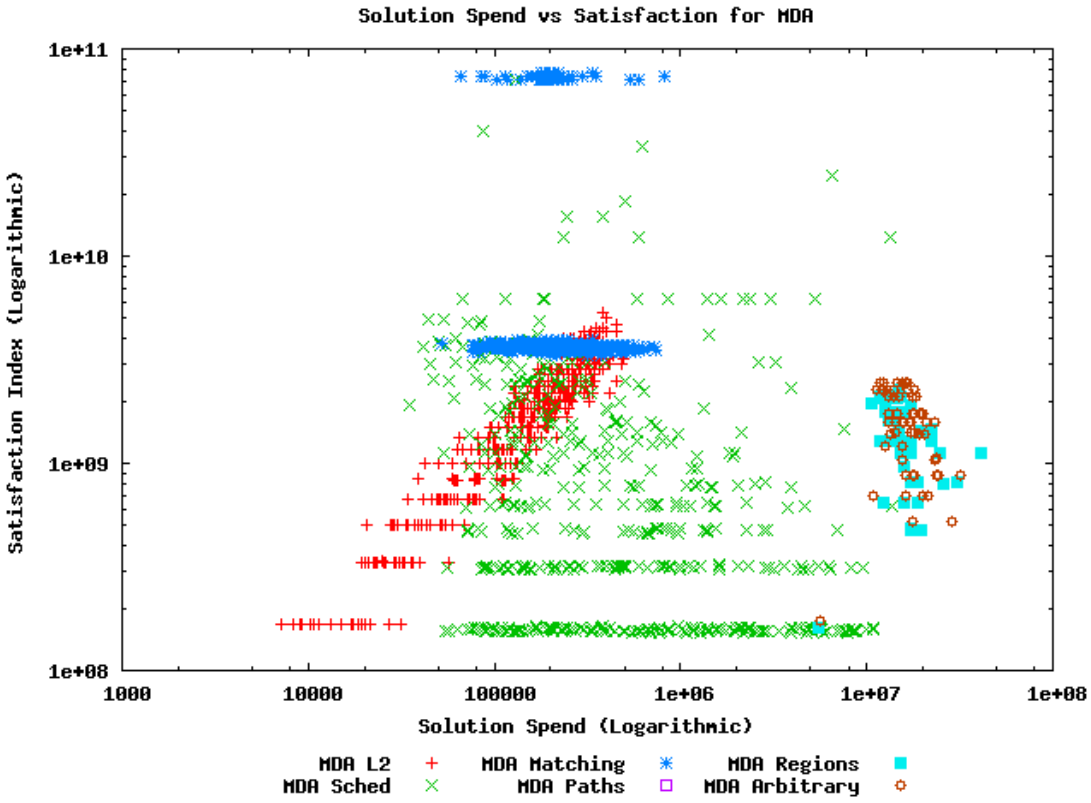

Figure 7.13: MDA Solution Spent vs Satisfaction

Chapter 8

Conclusions

8.1 Review of Contribution

Our objective has been to determine the most appropriate method to be used in order to compute the best allocation of a generic set of resources and to develop an understanding of how the different approaches available are related to one another.

Our motivation for this work stems from a need to make complex resource allocation decisions for distributed environments in a predicable amount of time, a motivation driven from needs we see illustrated in applications such as computer based Grid network management, response to train schedule disruptions and other multi-agent planning scenarios.

We also sought to examine the perceived wisdom of the combinatorial auction literature, in which many excellent theoretical papers have concluded that (almost to the extent that other mechanisms are not required) virtually all resource allocation problems can be solved using a centralised approach and that the NP-Hard characteristics of the combinatorial auctioneer are, for all practical purposes, mitigated through the use of careful development of heuristics and domain specific knowledge.

We believe that further advances in distributed computing and agent based technology will continue to present challenges for centralised approaches, both because of the impracticalities of encoding complete domain knowledge and perhaps more importantly,

we want to take advantage of the benefits of having a decentralised approach to decision making with no single point of command and control. Future systems will need the ability to perform useful resource allocations when only incomplete preferences and domain knowledge are available or forthcoming.

We developed an empirical analysis of a distributed market system and uniquely compared it to a centralised combinatorial auction solver "CASS" developed by Kevin Leyton-Brown in his thesis [LB03] using measures of hardness, financial, completion, efficiency, time and satisfaction, that are described in detail in Section 5.2.

Overall, we demonstrated that a distributed market provides an approach for solving more difficult problems, such as the job shop scheduling task simulated in our Scheduling distribution, and the distributed approach ensures that resource utilisation is greater than that in a centralised system, with good efficiency, revenue and decisions generated in a predicable time frame. Combinatorial solutions will always produce a Pareto efficient option and maximise revenue for the sellers of goods, but their run times vary unpredictably based on both the algorithm used and the complexity structure of the problem, and they often sacrifice resource utilisation.

8.2 Grounded in Economics

A further important component of this thesis is that the rationale and market structure for the decentralised mechanism is and should be, grounded in both Economic theory and the reality of market economics. Writing this conclusion in early 2009, one cannot but help observe that the real world experiences gained from living with markets has demonstrated that they demonstrate what happens when a large decentralised group of intelligent stock brokers utilise that uniquely human trait of making rational decisions about irrational behaviour and cause previously unforeseen, even unthinkable, waves of market disruption with massive knock-on effects, and the examples therein give a great opportunity for a thought experiment based evaluation of MDA.

8.2.1 Thought Experiment - Fiscal Markets

The financial system crash of 2009 is a good example of both the need for and the impact of decentralised markets. Previously, banks across the world lent money to each other on the basis that (i) at any given point in time, in respect of cash flow, some would be in surplus and some in deficit and (ii) banks with large cash deposits needed to ensure they gained maximum possible returns for their clients and lending to other secure banks who could lend on that money through products such as mortgages, loans etc. that paid a reasonable rate of interest was a sensible way to ensure maximum return.

In 2009 bankers began to understand that their colleagues had large and potentially undefinable risks associated with their lending portfolios and that the inter-bank loans may not be repaid. Suddenly banks were no longer considered "as safe as houses" and this myth was compounded when the US Government declined to bail out Lehman Brothers causing it to become bankrupt. Overnight there was a unanimous withdrawal from the wholesale lending markets as it was now seemingly impossible to trust that another bank was credit worthy and, as we know, the world's supply of credit evaporated over night.

This is a fascinating story, but what is the relevance to MDAs? Firstly, the wholesale credit market is made up of many autonomous players—banks, who join and exit the market as they please and where each has their own set of criteria, objectives and decision making processes. Secondly, there are many types of credit products, with different values, lengths, credit scores, etc. Thirdly, market operation is now a continuous process operating across multiple jurisdictions. The wholesale credit market is, therefore, very similar to our MDA and could be accurately modelled as one.

Could we use a series of Combinatorial Auctions to manage the wholesale credit market? Firstly, whilst it might be possible to run a continuous cycle of clearing operations as the number of products and participants grew non-linearly we would find issues with compute scalability and decision speed.

Secondly, consider the problem of ensuring all decisions were scrutinised by a single impartial auctioneer. There would be significant political issues of ensuring impartiality in a global system and it would also be logistically complex (impossible?) to have all participant revealing complete preference information about their purchase. The

logistics issue is worth further investigation. Whilst it is not beyond the bounds of technology for ten or even twenty thousand banks to send electronic data into a single reporting point it would be difficult for them to work out complete preference information without knowledge of the buying participants. Particularly because financial lending has significant risks attached, it is often the case that the price of the deal is easy to conclude and that the lenders view of the risk will improve during the negotiations as they increase familiarity with the buyer, a process which, in a market, would be implemented through a number of rounds of bidding and negotiation. In an MDA, both buyer and seller would be present in a single good-specific market with the potential for many rounds of one on one trading at which each can modify their position. In a combinatorial auction all buyers and sellers would need to express preferences centrally and await their outcome, a process that makes it impossible to express buyer specific preferences, or conduct rapid negotiations with specific buyers. Finally, the ability to alter preferences rapidly is of significant benefit, especially when third party interactions (in this case, Government bailouts and press speculation) will have a direct impact on lending availability and criteria.

Combinatorial Auctions do however have benefits in this scenario and there is significant attraction for the providers of credit to ensure that they receive the optimal allocation (and hence maximum profit) for their funds, but a very important criteria in financial lending is the social welfare factor understanding who the funds are sold to, which cannot be guaranteed in a blind auction.

8.2.2 MDA for real world problems

How then would the credit markets have fared in 2009 with the use of an MDA type mechanism? Well firstly, we can assert that they already do use a decentralised real-time set of continuous markets, likely in part due to the original design goals of our financial system that sought to protect us from systemic risk of having a centralised decision process and the difficulty in having a politically impartial auctioneer in a multi-national market. We believe that the flexibility of this structure did in some part facilitate the safe, but rapid withdrawal of parties from the trading arena when confidence collapsed and it has similarly allowed participants to re-enter on their own terms (timescales, risk, etc.).

With regards performance, we know that MDA would not have produced optimal value

for the sellers of funds but we demonstrated in Figure 6.4 that it would have facilitated an increase in the number of contracts sold, with a corresponding increase in our market efficiency measure. Note that most of the fees for sellers in fiscal markets are transactional and therefore related to volume, so potentially, earning more money from fees and less from interest is more beneficial to the sellers of this market. This is an interesting example of a scenario in which the producers of product who trade in the market have different objectives to the manufacturers of the goods they represent and highlights the difficulty of designing the correct market structure so that it incentivises the most desirable behaviour.

How would the system have worked if decision making was centralised? Assuming, given the constraints we have outlined, that it was indeed possible to build such a mechanism with good performance then a continuous cycle of combinatorial auctions would enable high quality lending to occur against well defined criteria. However there are two characteristics of the centralised market system that would potentially cause difficulty in the recent turbulent times. Firstly, a combinatorial auctioneer requires complete information about parties preferences in order to determine the optimal outcome and as mentioned, often negotiations for financial products require both parties to increase their preference revelation as the negotiation completes. Under a combinatorial auction mechanism, the auctioneer would attempt to bind all parties at each stage, thus denying each both the opportunity to revise their offers but also to adjust their preferences, for example about who they will deal with.

8.3 Economics provides the rationale

Returning to our discussion of desirable economic parameters from section 2.2, we see that our financial market requires us to take not only well defined concepts such as Pareto optimality into account when performing resource allocation but also the need for Social Welfare functions to be taken into account (in this case, the social welfare preferences for a banker would be trustworthiness and credit rating of the other party), and in this application context, these social welfare functions, and the ability to vary them rapidly, is potentially more important than some of the fiscal elements of the deal.

It is these fundamental economic characteristics of markets, such as Arrow's impossibility theorem [Arr50], a theorem which enables us to characterise social welfare and

gives a number of criteria which essentially shows that it is impossible to please everybody, that give us properties that mean a centralised combinatorial auction approach will not meet all of the future resource allocation needs and that an MDA approach to resource allocation is necessary for solving many of the example real world problems discussed from financial markets to train schedule exception handling.

We have shown, through a unique empirical analysis, that it will work and that whilst there are trade offs, most notably with regards the fiscal performance of the market, there are significant gains to be had from the use of a decentralised system with improved utilisation, throughput and time.

Whilst this is a Computer Science thesis, it is only when we set this work in the context of economic markets that we can truly understand the challenge that we will need to provide solutions for, with the next set of our computing technologies.

Chapter 9

Future Work and Directions

Our work has produced a comparison of two methods of performing bundle-oriented resource allocation with a view to understanding which approach is suitable for a given set of circumstances. This leads us to consider firstly, the approaches applicability for other, alternate situations and secondly, how we might improve our MDA implementation to maximise its utility in future.

9.1 Capability or Capacity?

Our objective in this thesis has been to answer the question: "Given a specific set of resources, allocation needs and network topology/geography (i.e centralised or decentralised) what is the most appropriate way to process the data and produce a set of resource allocations?" In the context of computer grid resource management, there is a supplementary question—Do we want capability or capacity computing?

Capability computing is defined as the provision of extremely high performance computing resources in order to handle the most difficult number crunching problems. Capacity computing deals with situations that have less extreme technical challenges but which require great quantities of computation resources.

The question posed above is still asked today by many users of large computing resources as we have not yet found a ubiquitous solution. Most large computing infras-

tructures are still built for specific purposes, whether they are simple AMD/Intel based clusters such as the University of Bath's Aquila system or most of the entrants on the Top500 list (e.g. IBM PowerCell or BlueGene systems and the Cray XT4 series). We see long commissioning times, leading edge processing developments and large budgets, with such systems usually seen installed in Universities, Research Centres and National Government computing facilities. The scheduling for large computing clusters is often done by hand, or using simple schedulers like PBS that divide jobs up on a first come first served basis, but the approach is problematic as it does not handle contention well with (i) an insufficient mix of capability and capacity users, (ii) management of resources at insufficient levels of granularity (e.g. operating in whole compute nodes) and (iii) failing to permit a very wide range of users concurrent access.

As an example of the problems seen on a low end cluster, the Aquila service is designed so that each job is assigned to one single CPU node (a quad-core processor with 8 GB of RAM). However, there are few nodes (approximately 80) and as a system used by many as their first interaction with cluster computing users find it difficult to program an algorithm that can continually exploit a full compute node. Therefore, the Aquila system rarely runs at full capacity and a review of usage on clusters such as the National Grid Service (a capacity service), or the HecTor Super Computer (a capability service) shows that achieving 100% utilisation can be difficult.

More commercially available clusters provide a different approach to the capacity computing need. Systems such as Amazon's EC2 system, RackSpace's Mosso or the 3Tera Applogic System allow users to purchase CPU hours on demand and operate clusters of virtual machines in the US and Europe. These work on a flat pricing basis but use service level controls to ensure that they can provision sufficient CPU units to cope with all possible demands.

We also find that the commercial clusters provide different approaches to service provision, allowing users to deploy entire virtual machines with custom software, where as most academically led clusters provide specific software running directly on physical computers optimised for the task. The former approach provides capacity computing, the latter focuses more on capability.

The mixture of controls, usage and over-provisioning of these systems has ensured that administrators have not yet discovered the need to implement just in time pric-

ing, however it is self evident that this will become necessary in the future. The move in the internet hosting industry towards virtualisation and cloud computing has abstracted the operating system environment from the computer's physical metal so that virtual machines can be moved around supported by hypervisors with resources reallocated as required. New versions of various opensource and commercial tools, including Xen, VMware and others are providing the ability for a service provider to deliver a virtualised computational resource service, where the user pays for what they use, rather than what they reserve. This just-in-time CPU resource delivery paradigm cannot infinitely be satisfied through provisioning large datacentres in the American mid-west and as globalisation continues to progress operators will begin to see significant peaks in capacity during business hours and the corresponding under usage overnight that will cause them to want to incentivise customers to spread their usage out more linearly. Similarly, the use of large amounts of computing power in academic and research circles will become normal, with undergraduate classes in physical and life sciences producing ever more complex models and whether the currency used is one with fiscal value or not, the need for a common metric for job scheduling and prioritisation cannot be under-estimated.

Our work has shown that the two approaches considered, the Combinatorial Auction and the Multiple Distributed Auction system are able to provide solutions to these challenges. The Combinatorial Auction, with its complete information about all preferences and single-shot allocation solver seems ideal for capability computing environments where workloads (particularly for super-computers) can be planned in advance and are not frequently changed. The MDA, on the other hand, is more suited to managing resources on capacity computing clusters, where loads change frequently with a continual stream of previously unknown jobs arriving and a much wider range of user and usage types.

In order to build and evaluate these two approaches to solving the problem of resource allocation in the thesis we necessarily focused on economic theories and the challenges of performing an empirical analysis. It is therefore appropriate to ask the question of how would we adapt this work so that it can be used in the scenarios described and what would be the areas of work to focus on next?

9.2 Other NP-Hard Problems

The D-CIS Lab at Thales has identified two areas of work which demonstrate applications of Agent systems being used to help NP-Hard problems in the real world, they call this paradigm "Actor-Agent Communities". Firstly, they have provided decision support software to help manage interruptions to train schedules in the Netherlands railway system and secondly to environmental impact decision making in Rotterdam Harbour.

Railway scheduling is a subject that has been long studied and with recent advances in computing power, heuristic development and search algorithms it is common place to find computer systems utilised in the production of initial railway schedules. However, the problem of managing the railway schedule in the event of a disruption to the network is currently unsolved. The Dutch railway system has over 5000 trains daily and 1000 driver duties per day, with an average of 10 disruptions per route, per day, equating to 450 trains a day with delays, of which 10 are cancelled.

The major issue is that time tables are typically developed to maximise usage of staff and rolling stock resources and that in the event of a blockage or disruption we find that drivers and their rolling stock are subsequently not available (in the right place at the right time) to begin the next service they are scheduled to operate. In most railway networks there is often surplus rolling stock and delivery of trains to support schedule interruptions is often not a great problem. Drivers however are in much shorter supply and it is rarely possible to arrange an entirely spare driver at short notice. Therefore, problems can only be resolved by rescheduling drivers, often swapping duties and requesting that overtime is worked. This work is currently completed by humans but researchers at D-CIS have observed that the human response often fixes the current problem but does not fully manage the knock on effects of the disruption, a situation which the D-CIS team have begun to address through their use of intelligent systems and prototype solutions have indicated that they can provide decision support tools to allow the problem to be resolved in a marginally faster time frame than that which was achieved by human planners, but additionally, they are able to ensure that the solution does not create knock on effects later in the schedule.

How could our work on multiple distributed auctions help further the developments of the D-CIS lab in refining their solution to this problem? Firstly, the train rescheduling

problem can be characterised as an NP-Hard problem with complex heuristics which requires search to produce a solution. However, when managing the crisis of train schedule interruption it is potentially more important to provide a workable solution quickly than to provide the optimum solution (i.e. one with zero knock on effects to the schedule) because a single knock on effect in the future can be easily managed, where as a current disruption may generate much larger, immediate effects in the network. Additionally, train drivers are known to be flexible and can be incentivised to implement sub-optimal solutions quickly, for example, through incentives in their pay and working conditions. Our work has shown that in applications where the optimal outcome is not necessary, a satisficing solution can be obtained through the use of MDAs and given further work it would be possible to identify the specific characteristics of different types of service interruptions such that the train scheduling tool could choose between using (i) previously memoised solutions, (ii) optimal rescheduling solutions, potentially determinable for small disruption problems, or (iii) a less optimal, but satisfactory solution, determined through market based agent negotiation.

The question of environmental decision support in Rotterdam Harbour is a problem to which we can similarly lend our expertise to improve the solution. The challenge in environmental management is the coordination of response across a very large number of different governmental agencies, all of whom have differing priorities and responsibilities, but whom must play a coordinated role in responding to an environmental accident. The Rotterdam Harbour presents an excellent case study in the management of these types of problems as it is one of the worlds busiest shipping environments, with a thriving petrochemical industry, that is geographically situated in close proximity to an area of high population density. The DIADEM system that the team at D-CIS have developed is an early prototype which begins to integrate the currently available sensor networks and human incident reports together with agent based logic to improve the distribution of information to the relevant authorities and enabling them to make higher quality decision in shorter times.

The MDA approach to resource allocation contains a number of concepts which are likely to be relevant to the future development of this project. Most importantly, given an element of information, or an identified need for action, there will be considerable coordination effort required across the many involved agencies. By representing the available agency resources and desired actions using the Trading Agents model we can effectively encourage teams to work together through the use of our bundling technology, recommending strategies for effective working which take into account and trade

off the many different constraints that will be imposed.

9.3 Review of the MDA System

We have outlined our implementation of a system of Multiple Distributed Auctions and the process of experiment and review has given us the opportunity to identify a number of areas where the system could be further improved.

We adopted a stepwise approach to re-factoring our Repast based simulation into the distributed Agentscape environment. The benefits of this approach were twofold, firstly, it was a fast process and secondly it allowed us to maintain the structural integrity of the market algorithms, thus ensuring that we had a valid basis for comparison.

The major downside is that the original implementation was built with a number of the Repast concepts deeply embedded, most significantly the "step" function and therefore the simulation was built on a cycle of (i) execute market, (ii) collect data. This approach has no impact on the market operation, but it massively increases the run time of the overall system because after each round of the market we sequentially poll all of the agents in the system to collect market data and statistics and in doing so, pause the market.

We propose that the efficiency of the MDA system could be greatly improved through modification so that statistical and accounting data is transmitted asynchronously to the Oracle outside of the regular trading process because then the markets could run continuously without interruption.

9.3.1 Intelligence of the Traders

The distributed market has a number of variables affecting its performance and unfortunately we could not thoroughly investigate them all but a potentially important factor is that of the intelligence and domain specific knowledge held by the market place traders. We focused on using Zero Intelligence traders using Dave Cliff's ZI-

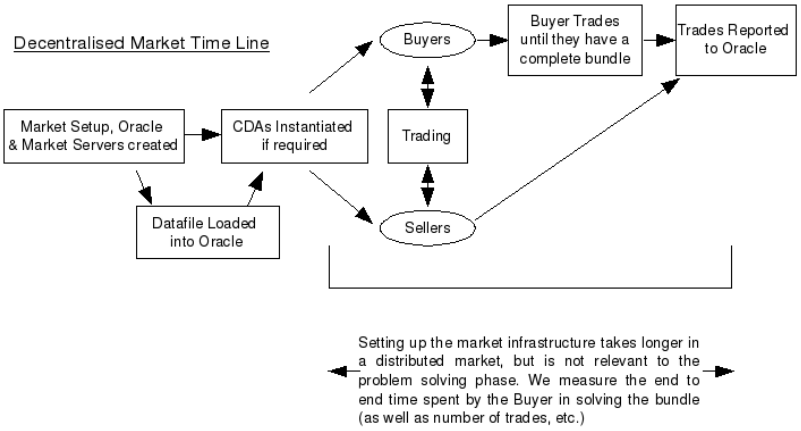

Figure 9.1: Break Down of Time Spent in Decentralised System

Plus strategy but work by participants in the Trading Agent Competition suggests that agents with domain specific expertise can produce economically superior results. For example, the MinneTAC agent participated in the Supply-Chain Trading Agent Competition (TAC-SCM). This competition involves participating in two concurrent games. Firstly, agents must purchase raw goods before, secondly, selling their finished bundles of goods. Participants in the game need to choose a strategy which typically involves reacting to one market or the other and being customer or supply driven. The MinneTac team developed a complex decision making process described as an "Evaluator Chain" which encapsulated the logic used in reacting to the different market events. They experimented with the two strategies and found that whilst the Customer-Driven strategy was more profitable overall, it had some significant downsides, such as inability to adapt to price fluctuations in the supply market. The conclusion is that agents in the game cannot adopt a single strategy and that balance is required—shifting priorities as markets change.

Additionally, awareness and knowledge of other strategies deployed in the market place, combined with a prediction model for customer demand was also found to be helpful. In computing, memoization is an optimization technique used primarily to speed up algorithms by remembering the answers to previously made function calls for common inputs, thus avoiding the need to repeat the calculation of results. We feel that this approach would allow traders to bid more quickly, with less computation required, on bundles for which they have bid on previously, providing particular performance gains as the markets increase in size from thousands, to tens of thousands of

goods.

9.3.2 Market Structure

The TAC-Market Design Competition (TAC-MDC) looked at the effect of a more intelligent market place, whilst maintaining fairly simple traders. In TAC-MDC the competition organisers managed the traders and used well known strategies for trading in continuous double auctions (ZI, ZIP, GD and RE (Roth-Erev [NPT01])) but they allowed participants to develop the continuous double auctions in which traders implemented by the competition organisers then took part in. The market operators made profits by charging traders transactional based fees for activities such as registering in the market, placing shouts, requesting information on other traders and for making bid/ask transactions. The competition organisers surveyed the techniques for the 2007 game in [NCP+08] and concluded that variations in the pricing policy, as well as the matching and shout acceptance policy, could improve the fees earned by market managers and the profitability of the market.

Our MDA system uses a simple free-entry continuous double auction which matches shouts on a first-come, first-served basis. We believe that given a more sophisticated market model, we should be able to influence the behaviour of the MDA system and further improve the market's economic efficiencies.

"Market Structure" encompasses both the economic makeup of the market, but also its linkages with the environment it controls and further work on MDA would need to be completed in order to make it compatible with the major capacity computing management systems Globus [Fos05] and GridBus [BB03]. In our MDA work we effectively traded goods through tokens that then enabled access to external providers of resources. In order to increase the future utility of MDA it would be beneficial to implement direct links between the MDA system and the relevant grid computing resource and accounting systems, such as the GridBus project "GridBank" and the Globus project GRAM environment. Both of these projects adopt a similar application programming interface based approach which allows third party products to be built as "pluggable modules" that can then connect to the Grid management system and provide support for allocation decisions and resource management.

9.4 Conclusions on Future Work

We believe that there is significant scope for the work done so that MDA can be utilised in a number of practical resource allocation scenarios, such as decision support and railroad planning and we have given examples for how this might be achieved, both structurally, with improvements to the MDA and MDA-Agentscape environments and further integration with capacity computing resource management systems but also at the application level by looking at specific applications which have strong applicability to the technology and ideas we have developed.

Central to the adoption of these ideas would also be the implementation of the identified improvements to the MDA and MDA-Agentscape, analysed in detail in Section 4.6 which will bring further improvements in speed, efficiency and security of the multiple distributed auctions system.

We are satisfied that the MDA system has the right balance between features and simplicity to enable us to fully investigate this thesis without it becoming a software engineering project. As the focus changed toward adoption, it would be important to ensure that a robust system for providing oversight was in place, principally to ensure traders acted within their financial means and did not hoard resources, or unnecessarily sell them at a loss.

Finally, the system works on a uniform trust model and there is potential for damage to be done if rogue agents were permitted to trade. We could increase the robustness and security frameworks in the system, for example, to include protocols and architectures derived from work by Franklin and Reiter [FR95] or Stubblebine and Syverson [SS99].

References

[ABC+09] Alvin AuYoung, Phil Buonadonna, Brent N. Chun, Chaki Ng, David C. Parkes, Jeff Shneidman, Alex C. Snoeren, and Amin Vahdat. Two auction-based resource allocation environments: Design and experience. In Rajmukar Buyya and Kris Bubendorfer, editors, Market Oriented Grid and Utility Computing, chapter 23. Wiley, 2009. (page 19, 19).

[AH71] Kenneth Arrow and F Hann. General Competitive Analysis. Published by North Holland, 1971. (pages 50, 121).

[AHDJ01] Patricia Anthony, W. Hall, Viet Dung Dang Dang, and Nicholas R. Jennings. Autonomous agents for participating in multiple online auctions. In Proc. of the IJCAI Workshop on EBusiness and the Intelligent Web, Seattle WA, USA, 2001. (pages 49, 71).

[AJ03] Patricia Anthony and Nicholas R. Jennings. Developing a bidding agent for multiple heterogeneous auctions. In ACM Trans on Internet Technology, volume 3, pages 185–217, New York, NY, USA, 2003. ACM. (page 49).

[Arr50] Kenneth J. Arrow. A difficulty in the concept of social welfare. The Journal of Political Economy, 58(4):328–346, 1950. (pages 50, 62, 153).

[ASGH95] David Abramson, Rok Sosic, J. Giddy, and B. Hall. Nimrod: A tool for performing parameterised simulations using distributed workstations. In HPDC, pages 112–121, 1995. (pages 15, 33, 46).

[ATY00] Arne Andersson, Mattias Tenhunen, and Fredrik Ygge. Integer programming for combinatorial auction winner determination.

Multi-Agent Systems, International Conference on, 0:0039, 2000. (page 133).

[Axe97] Robert M. Axelrod. The Complexity of Cooperation: Agent-Based Models of Competition and Collaboration. Princeton University Press, 1997. (page 52).

[BB03] A. Barmouta and Rajkumar Buyya. Gridbank: a grid accounting services architecture (gasa) for distributed systems sharing and integration. In Parallel and Distributed Processing Symposium, 2003. Proceedings. International, pages 8 pp.+, 2003. (page 162).

[BBP] Rachel A. Bourne, John Bigham, and Stefan Poslad. Effect of market type in continuous environments: The need for intermediaries. (page 40).

[BCC+01] Francine Berman, Andrew Chien, Keith Cooper, Jack Dongarra, Ian Foster, Dennis Gannon, Lennart Johnsson, Ken Kennedy, Carl Kesselman, John Mellor-Crummey, Dan Reed, Linda Torczon, and Rich Wolski. The GrADS Project: Software support for high-level Grid application development. The International Journal of High Performance Computing Applications, 15(4):327–344, 2001. Available via `http://www.cs.rice.edu/~johnmc/papers/GrADS-JHPCA-01.pdf`. (pages 44, 45).

[Ber38] Abram Bergson. A reformulation of certain aspects of welfare economics. The Quarterly Journal of Economics, 52(2):310–334, 1938. (pages 50, 61).

[BGS99] Craig Boutilier, Moisés Goldszmidt, and Bikash Sabata. Sequential auctions for the allocation of resources with complementarities. In IJCAI '99: Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, pages 527–523. Morgan Kaufmann Publishers Inc., 1999. (page 133).

[BKL03] Martin Bichler, Jayant R. Kalagnanam, and Ho Soo Lee. Reco: Representation and evaluation of configurable offers. In In Computational Modeling and Problem Solving in the Networked World: Interfaces in Computing and Optimization. Kluwer Academic Publishers, 2003. (page 40).

[BKR98] Jonathan Bredin, David Kotz, and Daniela Rus. Utility driven mobile-agent scheduling. Technical Report PCS-TR98-331, Dartmouth College, 1998. Available via http://www.cs.dartmouth.edu/reports/abstracts/TR98-331/ 24-03-2009. (page 39).

[BN05] Shantanu Biswas and Y. Narahari. Iterative dutch combinatorial auctions. Annals of Mathematics and Artificial Intelligence, 44(3):185–205, 2005. (pages 34, 52, 62).

[BPR99] Fabio Bellifemine, Agostino Poggi, and Giovanni Rimassa. JADE - a FIPA-compliant agent framework. In Proceedings of the Practical Applications of Intelligent Agents, 1999. (page 41).

[Bro04] Reva Brown. Consideration of the origin of herbert simon's theory of "satisficing" (1933-1947). Management Decision, 42(10):1240–1256, 2004. (page 57).

[Buy02] Rajkumar Buyya. Economic-based Distributed Resource Management and Scheduling for Grid Computing. PhD thesis, Monash University, Melbourne, Australia, 2002. Available via http://www.buyya.com/thesis/ 25-04-2009. (page 44).

[BW96] Francine Berman and Richard Wolski. Scheduling from the perspective of the application. In HPDC '96: Proceedings of the 5th IEEE International Symposium on High Performance Distributed Computing, page 100, Washington, DC, USA, 1996. IEEE Computer Society. (page 43).

[BW97] Francine Berman and Richard Wolski. The apples project: A status report. In 8th NEC Research Symposium, Berlin, Germany, May 1997, 1997. (page 43).

[CB98] Dav Cliff and Janet Bruten. Zero is not enough: On the lower limit of agent intelligence for continuous double auction markets. Tech. Rep. no. HPL-97-141, Hewlett-Packard Laboratories., 1998. (pages 49, 70, 81).

[CC02] Brent N. Chun and David E. Culler. User-centric performance analysis of market-based cluster batch schedulers. In CCGRID '02: Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid, page 30, Washington, DC, USA, 2002. IEEE Computer Society. (page 39).

[CDE+06] Yann Chevaleyre, Paul E. Dunne, Ulle Endriss, Jérôme Lang, Michel Lemaître, Nicolas Maudet, Julian Padget, Steve Phelps, Juan A. Rodríguez Aguilar, and Paulo Sousa. Issues in multiagent resource allocation. Informatica, 30:3–31, 2006. (pages 32, 50).

[CELM07] Yann Chevaleyre, Ulle Endriss, Jérôme Lang, and Nicolas Maudet. A short introduction to computational social choice. In Proceedings of the 33rd Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM-2007), volume 4362 of LNCS, pages 51–69. Springer-Verlag, January 2007. (page 51).

[CJS+02a] Junwei Cao, S. Jarvis, D. Spooner, J. Turner, D. Kerbyson, and G. Nudd. Performance prediction technology for agent-based resource management in grid environments. In 16th International Parallel and Distributed Processing Symposium (IPDPS '02 (IPPS and SPDP)), page 86. IEEE, April 2002. Available from `http://www.dcs.warwick.ac.uk/~hpsg/html/downloads/public/docs/CaoJ.PPTARM.pdf` 25-04-09. (page 34).

[CJS+02b] Junwei Cao, Stephen A. Jarvis, Subhash Saini, et al. ARMS: An agent-based resource management system for grid computing. Scientific Programming, 10(2):135–148, 2002. Available from `http://www.dcs.warwick.ac.uk/~saj/papers/arms.pdf` 25-04-2009. (pages 34, 53).

[CKN01] Junwei Cao, Darren J. Kerbyson, and Graham R. Nudd. Use of agent-based service discovery for resource management in metacomputing environment. Lecture Notes in Computer Science, 2150:882–886, 2001. (page 34).

[Cli97] Dave Cliff. Minimal-intelligence agents for bargaining behaviors in market-based environments. Technical Report HP–97–91, Hewlett Packard Laboratories, Bristol, England, 1997. `http://www.hpl.hp.com/techreports/97/HPL-97-91.html` 25-04-2009. (pages 49, 81).

[CSS05] Peter Cramton, Yoav Shoham, and Richard Steinberg, editors. Combinatorial Auctions. MIT Press, 2005. ISBN: 0-262-03342-9. (pages 34, 58, 63, 64, 64, 133, 135).

[DF03] Li Ding and Tim Finin. Strategies and heuristics used by the umbctac agent. In IJCAI-03 Workshop on Trading Agent Design and Analysis, Acapulco, 2003. http://www.cs.umbc.edu/~finin/papers/ijcai03-umbctac.pdf 25-04-2009. (page 71).

[DJ03] Viet Dung Dang and Nicholas R. Jennings. Optimal clearing algorithms for multi-unit single-item and multi-unit combinatorial auctions with demand/supply function bidding. In ICEC '03: Proceedings of the 5th international conference on Electronic commerce, pages 25–30. ACM Press, 2003. (pages 24, 35, 52, 66, 71).

[dVV03] S. de Vries and R. V. Vohra. Combinatorial auctions: A survey. INFORMS Journal on Computing, 15(3):284–309, 2003. (page 133).

[ERS+05] Torsten Eymann, Michael Reinicke, Werner Streitberger, Omer Rana, Liviu Joita, Dirk Neumann, Björn Schnizler, Daniel Veit, Oscar Ardaiz, Pablo Chacin, Isaac Chao, Felix Freitag, Leandro Navarro, Michele Catalano, Mauro Gallegati, Gianfranco Giulioni, Ruben Carvajal Schiaffino, and Floriano Zini. Catallaxy-based grid markets. In Multiagent and Grid Systems, volume 1, pages 297–307. IOS Press, 2005. (page 38).

[FBK96] Stephen J. Fink, Scott B. Baden, and Scott R. Kohn. Flexible communication mechanisms for dynamic structured applications. In IRREGULAR '96: Proceedings of the Third International Workshop on Parallel Algorithms for Irregularly Structured Problems, pages 203–215, London, UK, 1996. Springer-Verlag. (pages 15, 46).

[FLBS99] Yuzo Fujishima, Kevin Leyton-Brown, and Yoav Shoham. Taming the computational complexity of combinatorial auctions: Optimal and approximate approaches. In IJCAI '99: Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, pages 548–553, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc. (page 133).

[Fos05] Ian T. Foster. Globus toolkit version 4: Software for service-oriented systems. In Hai Jin, Daniel A. Reed, and Wenbin Jiang, editors, NPC, volume 3779 of Lecture Notes in Computer Science, pages 2–13. Springer, 2005. (page 162).

[FR95] M. Franklin and M. Reiter. The Design and Implementation of a Secure Auction Service. In Proc. IEEE Symp. on Security and Privacy, pages 2–14, Oakland, Ca, 1995. IEEE Computer Society Press. (pages 41, 73, 163).

[GK94] Michael R. Genesereth and Steven P. Ketchpel. Software agents. Communications of the ACM, 37:48–53, 1994. (page 87).

[GO96] Russell Greiner and Pekka Orponen. Probably approximately optimal satisficing strategies. Artificial Intelligence, 82(1–2):21–44, April 1996. (page 52).

[GOT+08] Peter Gradwell, Michel Oey, Reinier Timmer, Frances Brazier, and Julian Padget. Engineering large-scale distributed auctions. In AAMAS '08: Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems, pages 1311–1314, Richland, SC, 2008. International Foundation for Autonomous Agents and Multiagent Systems. (pages 29, 34).

[GP05] Peter Gradwell and Julian Padget. Markets vs auctions: Approaches to distributed combinatorial resource scheduling. In Multiagent and Grid Systems, number 4/2005 in 1, pages 251–262. IOS Press, 2005. (page 31).

[GP07] Peter Gradwell and Julian Padget. A comparison of distributed and centralised agent based bundling systems. In ICEC '07: Proceedings of the ninth international conference on Electronic commerce, pages 25–34, New York, NY, USA, 2007. ACM Press. (pages 30, 110).

[GS93] Dhananjay K. Gode and Shyam Sunder. Allocative efficiency of markets with zero-intelligence traders: Market as a partial substitute for individual rationality. In The Journal of Political Economy, volume 101, pages 119–137. The University of Chicago Press, Feburary 1993. (page 81).

[HJ02] Minghua He and Nicholas R. Jennings. Southamptontac: Designing a successful trading agent. In 15th European Conf. on AI (ECAI-2002), pages 8–12, 2002. (pages 48, 71).

[KCG+07a] Wolfgang Ketter, John Collins, Maria Gini, Alok Gupta, and Paul Schrater. Detecting and forecasting economic regimes in automated

exchanges. Technical Report 07-008, University of Minnesota, Dept of Computer Scienceand Engineering, Minneapolis, MN, 2007. (page 48).

[KCG+07b] Wolfgang Ketter, John Collins, Maria Gini, Paul Schrater, and Alok Gupta. A predictive empirical model for pricing and resource allocation decisions. In ICEC '07: Proceedings of the ninth international conference on Electronic commerce, pages 449–458, New York, NY, USA, 2007. ACM. (page 48).

[KKD+04] Wolfgang Ketter, Elena Kryzhnyaya, Steven Damer, Colin McMillen, Amrudin Agovic, John Collins, and Maria L. Gini. Analysis and design of supply-driven strategies in tac scm. In TADA Workshop 2004, AAMAS Conference, 2004. (page 71).

[KS89] James F Kurose and Rahul Simha. A microeconomic approach to optimal resource allocation in distributed computer systems. In IEEE Transactions On Computers Vol 38, pages 705–716, 1989. (pages 15, 46).

[LB03] Kevin Leyton-Brown. Resource Allocation in Competitive Multiagent Systems. PhD thesis, Stanford University, August 2003. (pages 21, 24, 65, 71, 133, 150).

[LBNS02] Kevin Leyton-Brown, Eugene Nudelman, and Yoav Shoham. Learning the empirical hardness of optimization problems: The case of combinatorial auctions. In Constraint Programming (CP), pages 556–572, 2002. (pages 104, 105).

[LBNS06] Kevin Leyton-Brown, Eugene Nudelman, and Yoav Shoham. Empirical Hardness Models for Combinatorial Auctions, chapter 19. MIT Press, 2006. (page 110).

[LBPS00] Kevin Leyton-Brown, Mark Pearson, and Yoav Shoham. Towards a universal test suite for combinatorial auction algorithms. In EC '00: Proceedings of the 2nd ACM conference on Electronic commerce, pages 66–76, New York, NY, USA, 2000. ACM. (pages 19, 25, 76, 103).

[LBS05] Kevin Leyton-Brown and Yoav Shoham. A Test Suite for Combinatorial Auctions, chapter 18, pages 452–478. MIT Press, 2005. (pages 101, 109).

[LJC+08] Benjamin Lubin, Adam Juda, Ruggiero Cavallo, Sébastien Lahaie, Jeffrey Shneidman, and David C. Parkes. ICE: An Expressive Iterative Combinatorial Exchange. Journal of Artificial Intelligence Research, 33:33–77, 2008. (page 19).

[LKK99] William Leinberger, George Karypis, and Vipin Kumar. Job scheduling in the presence of multiple resource requirements. In Supercomputing '99: Proceedings of the 1999 ACM/IEEE conference on Supercomputing (CDROM), page 47. ACM, 1999. (page 45, 45).

[LSP82] Leslie Lamport, Robert Shostak, and Marshall Pease. The Byzantine Generals Problem. ACM Transactions on Programming Languages and Systems, 4(3):382–401, July 1982. (page 42).

[Mai05] Roger Mailler. Comparing two approaches to dynamic, distributed constraint satisfaction. In AAMAS 2005: Proceedings of the 4th International Conference on Autonomous Agents and Multi-Agent Systems, pages 1049–1056, 2005. (page 71).

[Mic68] Donald Michie. Memo functions and machine learning. Nature, 218, April 1968. (page 73).

[MT04] ToddWright Aaron Helsinger Michael Thome. Cougaar: A scalable, distributed multi-agent architecture,. In IEEE International Conference on Systems, Man and Cybernetics, pages 1910– 1917, 2004. (page 41).

[Nas50] John F. Nash. Equilibrium points in n-person games. Proceedings of the National Academy of Sciences of the United States of America, 36(1):48–49, 1950. (page 56).

[NCP+08] Jinzhong Niu, Kai Cai, Simon Parsons, Enrico Gerding, and Peter McBurney. Characterizing effective auction mechanisms: insights from the 2007 tac market design competition. In AAMAS '08: Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems, pages 1079–1086. International Foundation for Autonomous Agents and Multiagent Systems, 2008. (page 162).

[NCV06] Michael J. North, Nicholson T. Collier, and Jerry R. Vos. Experiences creating three implementations of the repast agent modeling toolkit. ACM Trans. Model. Comput. Simul., 16(1):1–25, 2006. (pages 19, 40, 81).

[NKP+00] G. R. Nudd, D. J. Kerbyson, E. Papaefstathiou, S. C. Perry, J. S. Harper, and D. V. Wilcox. Pace–a toolset for the performance prediction of parallel and distributed systems. Int. J. High Perform. Comput. Appl., 14(3):228–251, 2000. (page 34).

[NM09] Cynthia Nikolai and Gregory Madey. Tools of the trade: A survey of various agent based modeling platforms. Journal of Artificial Societies and Social Simulation, 12(2):2, 2009. (page 41).

[NPT01] James Nicolaisen, Valentin Petrov, and Leigh Tesfatsion. Market power and efficiency in a computational electricity market with discriminatory double-auction pricing. IEEE Transactions on Evolutionary Computation, 5:504–523, 2001. (page 162).

[OB06] B. J. Overeinder and F. M. T. Brazier. Scalable middleware environment for agent-based Internet applications. In Applied Parallel Computing, volume 3732 of Lecture Notes in Computer Science, pages 675–679. Springer, Berlin, 2006. (pages 25, 40, 86, 86).

[Par03] David Parkes. Computational mechanism design - taming the strategic dragon without invoking the complexity monster, 2003. Talk given at AAMAS03. (page 37).

[PB99] Julian Padget and Russell Bradford. A π-calculus model of the spanish fishmarket. In Proceedings of AMET'98, volume 1571 of Lecture Notes in Artificial Intelligence, pages 166–188. Springer Verlag, 1999. (page 42).

[PMPM06] S. Phelps, M. Marcinkewitz, S. Parsons, and P. McBurney. A novel method for strategy acquisition in n-player games. Fifth International Join Conference on Automous Agents and Multiagent Systems, 2006. `http://www.csc.liv.ac.uk/~sphelps/docs/pubs/aamas06.ps.gz` 25-04-2009. (pages 19, 72, 80).

[Pro] Globus Project. `http://www.globus.org/` Last accessed March 2005. (page 43).

[PS04] David C. Parkes and Jeffrey Shneidman. Distributed implementations of vickrey-clarke-groves mechanisms. In Proc. 3rd Int. Joint Conf. on Autonomous Agents and Multi Agent Systems, pages 261–268, 2004. (pages 19, 43, 66).

[RLJ06] Steven F. Railsback, Steven L. Lytinen, and Stephen K. Jackson. Agent-based Simulation Platforms: Review and Development Recommendations. SIMULATION, 82(9):609–623, 2006. (page 41).

[RNSP97] Joan-Antoní Rodríguez, Pablo Noriega, Carles Sierra, and Julian Padget. Fm96.5 a java-based electronic auction house. In Proceedings of 2nd Conference on Practical Applications of Intelligent Agents and MultiAgent Technology (PAAM'97), pages 207–224, London, UK, April 1997. ISBN 0-9525554-6-8. (page 42).

[SAL+04] Jahanzeb Sherwani, Nosheen Ali, Nausheen Lotia, Zahra Hayat, and Rajkumar Buyya. Libra: a computational economy-based job scheduling system for clusters. Softw., Pract. Exper., 34(6):573–590, 2004. (page 44, 44).

[San02] Tuomas Sandholm. Algorithm for optimal winner determination in combinatorial auctions. Artif. Intell., 135(1-2):1–54, 2002. (pages 35, 36, 133).

[San06] Tuomas Sandholm. Optimal winner determination algorithms. In Peter Cramton, Yoav Shoham, and Richard Steinberg, editors, Combinatorial Auctions, chapter 14. MIT Press, 2006. (page 36).

[San07] Tuomas Sandholm. Expressive commerce and its application to sourcing: how we conducted $35 billion of generalized combinatorial auctions. In ICEC '07: Proceedings of the ninth international conference on Electronic commerce, pages 349–350, New York, NY, USA, 2007. ACM Press. (pages 35, 65).

[SCG07] Eric Sodomka, John Collins, and Maria L. Gini. Efficient statistical methods for evaluating trading agent performance. In AAAI, pages 770–775. AAAI Press, 2007. (pages 34, 49).

[SG06] Tuomas Sandholm and Andrew Gilpin. Sequences of take-it-or-leave-it offers: near-optimal auctions without full valuation revelation. In AAMAS '06: Proceedings of the fifth international joint conference

on Autonomous agents and multiagent systems, pages 1127–1134, New York, NY, USA, 2006. ACM. (page 18).

[SH80] Rajan Suri and Yu-Chi Ho. Resource management for large systems: Concepts, algorithms, and an application. In IEEE Transactions On Automatic Control Vol 4, pages 651–662, 1980. (pages 15, 46).

[Sim55] Herbert A. Simon. A behavioral model of rational choice. The Quarterly Journal of Economics, 69(1):99–118, 1955. (pages 18, 23, 37, 50, 57, 57).

[Sma76] Stephen Smale. Dynamics in general equilibrium theory. American Economic Review, 66(2):288–94, May 1976. (page 36).

[SP03] Jeffrey Shneidman and David C. Parkes. Rationality and self-interest in peer to peer networks. In In 2nd Int. Workshop on Peer-to-Peer Systems (IPTPS03, pages 139–148, 2003. (pages 37, 38, 42).

[SP04] Jeffrey Shneidman and David C. Parkes. Specification faithfulness in networks with rational nodes. In PODC '04: Proceedings of the twenty-third annual ACM symposium on Principles of distributed computing, pages 88–97, New York, NY, USA, 2004. ACM. (pages 37, 42, 42).

[SS99] Stuart G. Stubblebine and Paul F. Syverson. Fair on-line auctions without special trusted parties. In FC '99: Proceedings of the Third International Conference on Financial Cryptography, pages 230–240, London, UK, 1999. Springer-Verlag. (pages 42, 73, 163).

[SSGL01] Tuomas Sandholm, Subhash Suri, Andrew Gilpin, and David Levine. CABOB: A fast optimal algorithm for combinatorial auctions. In IJCAI, pages 1102–1108, 2001. (page 35).

[SSGL05] Tuomas Sandholm, Subhash Suri, Andrew Gilpin, and David Levine. CABOB: A fast optimal algorithm for combinatorial auctions. In Management Science, Special Issue on Electronic Markets, 2005. (pages 35, 106).

[TDTY04] Rui Dong Terry, Rui Dong, Terry Tai, and Wilfred Yeung. Hartac - the harvard tac scm '03 agent. In Proceedings of the Trading Agent Design and Analysis (TADA2004) Workshop, 2004. MAMS-PG 6.58 MAMS-PG-SS 7.45 TestAgent, pages 1–8, 2004. (page 71).

[TH04] Robert Tobias and Carole Hofmann. Evaluation of free java-libraries for social-scientific agent based simulation. Journal of Artificial Societies and Social Simulation, 7, 2004. (page 41).

[TS08] Richard H. Thaler and Cass R. Sunstein. Nudge : improving decisions about health, wealth, and happiness. Yale University Press, 2008. (page 38).

[VD02] Sathish S. Vadhiyar and Jack J. Dongarra. A metascheduler for the grid. In HPDC '02: Proceedings of the 11th IEEE International Symposium on High Performance Distributed Computing, page 343. IEEE Computer Society, 2002. (page 45).

[vH76] Friedrich August von Hayek. Law, Legislation and Liberty - The Mirage of Social Justice, volume 2. University of Chicago Press, 1976. (page 38).

[WHH+92] Carl A. Waldspurger, Tad Hogg, Bernardo A. Huberman, Jeffrey O. Kephart, and W. Scott Stornetta. Spawn: A distributed computational economy. IEEE Trans. Softw. Eng., 18(2):103–117, 1992. (page 39).

[WPBB01] Rich Wolski, James S. Plank, John Brevik, and Todd Bryan. Analyzing market-based resource allocation strategies for the computational Grid. The International Journal of High Performance Computing Applications, 15(3):258–281, Fall 2001. (pages 24, 36, 36).

[WWWMM98] Michael P. Wellman, William E. Walsh, Peter R. Wurman, and Jeffrey K. MacKie-Mason. Auction protocols for decentralized scheduling. In Games and Economic Behaviour, volume 35, pages 271–303, 1998. (page 58).

[YB06] Chee Shin Yeo and Rajkumar Buyya. A taxonomy of market-based resource management systems for utility-driven cluster computing. Softw. Pract. Exper., 36(13):1381–1419, 2006. (page 44).

[ZFD+03a] Y. Zou, T. Finin, L. Ding, H. Chen, and R. Pan. Taga: Trading agent competition in agentcities. Draft submitted to the IJCAI Workshop on Trading Agent Design and Analysis, 2003. (page 47).

[ZFD+03b] Youyong Zou, Tim Finin, Li Ding, Harry Chen, and Rong Pan. Using semantic web technology in multi-agent systems: a case study in the

taga trading agent environment. In ICEC '03: Proceedings of the 5th international conference on Electronic commerce, pages 95–101, New York, NY, USA, 2003. ACM. (page 47).