ABSTRACT

PRITI P PHADKE. An Evolutionary Approach to finding Bidding Strategies in a Combinatorial Auction. (Under the direction of Dr. Peter Wurman.)

Auctions involve trading of variety of different items. Auctions that allow agents to bid for combinations of items are called Combinatorial Auctions (CAs). The Ascending k-Bundle Auction(AkBA) is a combinatorial auction founded on a notion of bundle price equilibrium. The purpose of this research is to explore the strategy space and help agents evolve strategies for a Proxy version of A1BA(P-AkBA). We use a Genetic algorithm to search the space of strategies. Several experiments were performed for different categories of problems and the results show that the approach yields good solutions. We compare the outcomes of the evolved solutions with the outcomes that result from truthful bidding, and compare prices against those generated in the sealed-bid version of k-bundle auction and the standard GVA payments. We also make several observations about the effect of genetic parameters on the performance of search.

AN EVOLUTIONARY APPROACH TO FINDING BIDDING STRATEGIES IN A COMBINATORIAL AUCTION

by

Priti P Phadke

A thesis submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Master of Science

Computer Science

Raleigh

July 2002

APPROVED BY:

Chair of Advisory Committee

To dear Aai and Baba

BIOGRAPHY

Priti Phadke was born in Pune, India in 1978. She grew up in Bombay, where she also got her Bachelors in Computer Engineering from Bombay University in 1999. After working for a year as a software engineer, she joined NC State in Fall 2000 for the Masters program in Computer Science.

ACKNOWLEDGEMENTS

I take this opportunity to thank all those people without whom I would not have been able to realize my Masters degree and this thesis in particular.

I would like to thank my advisor, Dr.Peter Wurman for his direction, guidance and support. He has been an excellent guide, and his enthusiasm about work is really contagious. I have thoroughly enjoyed the learning experience during these two years, and I am glad I got the opportunity to work with him. Special thanks to Dr.Jon Doyle, his comments about our project helped us a lot. I thank him and Dr.Michael Young for participating in my thesis committee.

Special thanks to Karthik, for being such a wonderful friend, and his help during the writing of this thesis. Thanks to Sonali, for all the necessary diversions, and good lunch breaks. Thanks to Harshit and Harish who have been helpful in many ways.

My parents deserve a big share in my success, and I thank them for having faith in me and always stressing on the importance of education. Thanks Mom and Jo for writing those long, encouraging letters. Thanks Dad for everything, you are the reason I am here.

Finally, words alone cannot express the thanks I owe to Rahul, for all the love, support and encouragement.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Auctions are one of the oldest business models in history. Auctions date at least as early as 193 AD, when Didius purchased the Roman emperor's crown for 6250 drachmas [4]. Today, with perhaps less glamor but with a large dollar volume, the United States Treasury auctions off its treasury bills and bonds.[1] Internet auctions on Ebay, Yahoo!, Amazon.com enable trade of millions of items every week. The Federal Communications Commission (FCC) uses auctions to sell segments of radio frequency spectrum to telecommunications companies [15].

Auctions are popular because they enable dynamic pricing. They are especially useful in selling a commodity of undetermined market value; the demand and supply in the market determines the value of the item being sold. Many trading scenarios involve the sale of a variety of different items. Participants in the auction may have preferences for combinations of those items; resulting in complementarities (or substitution effects) between different items [8]. Auctions that allow participants to bid on combinations of items are called *Combinatorial Auctions (CA)*.

Consider the following example. Alice's hard disk has bad sectors, and she wants to buy a new hard disk. She has a very old motherboard, and it does not support the latest 20GB hard disks that are available. As a result, she has to buy a motherboard

[1] For the source of these and other anecdotes, see [6].

too. She values both the motherboard and the hard disk for a value $v(AB)$, but has little use for just one of the two items alone. If $v(A)$ and $v(B)$ are the values for the hard disk and motherboard individually, then in this case $v(AB) > v(A)+v(B)$. Here, Alice exhibits *complementary preferences* represented by the superadditive function. In the extreme case Alice might have, $v(AB) >> 0$ and $v(A) = v(B) = 0$, where she has no value for the individual item, and values only the combination. *Substitutable preferences* exist when the buyer values an item less in conjunction with another item. For example, a person who wants to buy a raincoat, would have substitutable preferences for different colored raincoats. If $v(A)$ is the buyer's value for a red raincoat and $v(B)$ is the value for a blue raincoat, then in this case $v(AB) < v(A) + v(B)$. In the extreme case, the buyer may have a preference of $v(AB) = max[v(A), v(B)]$. We assume that all items are distinguishable from each other in this case; and that agents can freely dispose of items they do not want.

We focus on combinatorial auctions in our research.

1.2 Motivation

Auctions can be viewed as a form of negotiation, where buyer(s) and seller(s) strike a deal on a mutually agreed issue: price. Negotiation can be viewed as a search process [20]. When two players are bargaining, they can actually be viewed as two negotiators jointly searching a multi-dimensional space and mutually agreeing to a single point in space.

Each buyer (also referred to as a bidder or an agent) participating in an auction, has a *strategy* that he uses to bid in the auction. A strategy is the buyer's plan that defines how much to bid, on what bundle to bid, and when to bid, based on the information made available by the auction. It could be the set of values for the items that the bidder wants to bid in the auction. The set of all possible strategies for all the bidders form a very large multi-dimensional space. In a combinatorial auction, where bidders have complementary or substitutable preferences, it is necessary that they have some insight on the space of strategies available to be able to pick the best one. Each buyer's action is dependent on other buyers' actions and hence buyers

have to implicitly cooperate to agree on a solution. Such solutions are said to be in *equilibrium (Nash)*, where no buyer wants to unilaterally deviate from his given strategy.

It is necessary to devise a way of finding such good equilibrium solutions. However, the size of the strategy space is very large, even for two agents, two-item problem with many local optimum solutions and multiple Nash equilibria. Also, due to the complex nature of the problems, the strategy space is known not to be perfectly smooth and unimodal (i.e., a single smooth slope upwards or downwards).

Due to these complexities, it is difficult to explore the entire space in all dimensions, thus our goal in this thesis is to evolve reasonable strategies for a combinatorial auction. To work towards this goal, we have developed a program that uses a genetic algorithm that evolves a bidding strategy.

1.3 Organization of thesis

Chapter 2 is a review of the issues in Combinatorial Auctions, the Generalized Vickery Auction and the Ascending k-bundle auction. In Chapter 3, I discuss the need to use a genetic algorithm, and introduce genetic algorithms in general. Chapter 4 describes the evolutionary approach we use. The design of the experiments is explained in Chapter 5. Chapters 6 and 7 discuss the results, conclusions and future work.

Chapter 2

Combinatorial Auctions

Auctions have been classified as ascending/descending, sealed-bid/out-cry, single/double or based on the rules for bidding and payments. No matter what the rules are, all auctions have a few activities in common. Wurman, Wellman and Walsh [32] define the following three core activities common to auctions:

- Receive bids: Bids are messages by the agents to indicate their willingness to buy the items on which bids are placed. The auctioneer admits the bids after verifying that they conform to the rules of the auction.

- Clear: This step determines the resource allocation and the payments between the buyers and sellers. It consists of two tasks: *Winner Determination* and *Payment Determination*. Winner determination is the process of determining which bidders will trade and at what quantities. Payment determination is the task of setting the price for each trade.

- Reveal intermediate information: This is the intermediate information that the auction supplies to the agents. It is usually the status of the auction at that moment with respect to the allocation and payments.

The first two steps are mandatory to all auctions. Revealing intermediate information is optional; if the auction is *sealed-bid* no intermediate information is revealed. Each auction has its own way of conducting these core activities. An auction may

interleave and iterate over these steps any number of times, depending on the rules of the auction.

Combinatorial Auctions can be classified based on whether they are sealed-bid or ascending with multiple rounds. There are several variations in setting of the rules for submitting bids, deciding the allocation, and computing the payments. The seminal work of Vickrey [29] and its extension to problems with bundles of items, provides the theoretical outline of combinatorial auctions.

2.1 General Model

Consider a set J of m items indexed by j. Let B denote all $(2^m - 1)$ (excluding the null set) possible subsets of J which we refer to as *bundles*, indexed by b. Let b^j indicate that item j is an element of b. Consider a set I of n agents indexed by i.

Agent i has valuations $v_i(b)$ for the bundles $b \in B$. If $b \supset c$, then $v_i(b) \geq v_i(c)$, this assumption called free disposal, allows us to assume that values increase monotonically. Let $x_{ib} \in \{0, 1\}$, where $x_{ib} = 1$ iff agent i receives bundle b, and 0 otherwise. The solution to the Combinatorial Allocation Problem (CAP) is the best possible assignment of bundles to agents. This is also known as the *Winner Determination Problem (WDP)*. The value of the socially efficient allocation is given by,

$$\begin{aligned} \max \quad & \sum_i \sum_b v_i(b) x_{ib} \\ \text{s.t} \quad & \sum_b x_{ib} \leq 1 \text{ i} = 1 \ , \ldots , \text{ m}, \\ & \sum_i \sum_b b^j x_{ib} \leq 1, j = 1, \ldots, n, \\ & x_{ib} \in \{0, 1\} \end{aligned}$$

The constraints specify that each agent receives at most one bundle and that no item is allocated more than once. Let f^* be the solution to this integer problem. Let V(f^*) be the value of the solution f^*.

In our research, we focus on an ascending combinatorial auction called the Ascending k-Bundle Auction (AkBA) [30]. We compare our results with the Generalized

Agent	A	B	C	AB	AC	BC	ABC
1	2	5	5	7	5	7	15
2	2	1	8	2	9*	8	10
3	6	10*	3	10	7	13	18

Table 2.1: Example Problem

Vickery Auction (GVA) [14] which can be viewed as an incentive compatible, sealed-bid Combinatorial Auction. In the following section, we review both the GVA and AkBA.

2.2 The GVA

The Generalized Vickery Auction [14], is a direct revelation mechanism, but is often treated as a sealed-bid combinatorial auction. Each participant reveals his utility function, that is, his valuation for all the bundles, to the auctioneer. The auctioneer computes the optimal allocation and the payment for each agent. The GVA is *incentive compatible*, a desirable property which ensures truthful bidding is a weakly dominant strategy.

2.2.1 Model

Each agent $i \in I$ submits a list of his valuations for all the bundles $b \in B$ to the auctioneer. Auctioneer solves the combinatorial allocation problem with these reported values and the solution is the value $V(f^*)$.

Consider the example in Table 2.1. The optimal allocation is denoted by an asterisk. In this case agents 2 and 3 are allocated bundles AC and B, respectively, which gives the maximum revenue.

2.2.2 GVA Payments

The GVA is an extension of the second price auction. In the second price auction, the highest bidder wins, but pays the amount equal to the second-highest bid (or

Agent	A	B	AB
1	5*	5	10
2	5	5*	8

Table 2.2: Discriminatory Prices Example for GVA

highest unsuccessful bid). The price that the winning bidder pays is determined by competitors' bids alone and does not depend upon any action the bidder undertakes.

To compute the prices for the GVA, the auctioneer has to solve the combinatorial allocation problem once to find the best allocation, and once again without each agent for computing the payments. The allocation and the utility of the auction if the agent was not there is computed as $V(f^{*(I\setminus i)})$. Each agent is charged the effect of his presence on the other agents. The payment for agent i is computed as,

$$\pi_i = V(f^{*(I\setminus i)}) - [V(f^*) - v_i(f_i^*)]$$

In the example in Table 2.1, the payments will be as follows:

- Agent 1: $19 - (19 - 0) = 0$
- Agent 2: $18 - (19 - 9) = 8$
- Agent 3: $15 - (19 - 10) = 6$

Each agent's bid can affect only his allocation. It does not change the payment, because the payment is dependent on other agents' valuations. Since the agent's payments do not depend on his valuations, truthful bidding is the dominant strategy in GVA.

The GVA payments are computed as a payment for each agent, not as the price for each object. Hence, there is no way to determine the relative price for the items in the allocated bundle. Also, GVA payments are *non-anonymous*, which means agents may pay different amounts for the same bundle if they win. Consider the example shown in Table 2.2. The payments in this example are non-anonymous, or *discriminatory*. Agent 1 pays \$3 and Agent 2 pays \$5 though they have the same value for the item. This can be undesirable, and in many situations, we expect participants will want uniform prices.

The GVA is an incentive compatible, efficient combinatorial auction. Though these properties make the GVA attractive, it has a high computational cost. It requires the agents to submit bids for all possible bundles, the specification of which can be computationally expensive. Also, the auctioneer has to solve the winner determination problem $(n + 1)$ times to determine the allocation and the payments for each agent. This motivates the need for a progressive combinatorial auction where agents need not compute the bids for all possible bundles at one time.

2.3 The AkBA

Ascending CAs iterate, allowing the agents to submit bids on bundles based on the feedback from the auction [21]. This can save a lot of computation for the agents. AkBA achieves the benefits of progressive mechanisms while also using anonymous, nonlinear pricing [30]. AkBA maintains anonymous prices and the prices are computed for each bundle; the agent who wins a bundle pays the price for the winning bundle. Prices in AkBA are always in equilibrium with respect to the bids.

The Ascending k-Bundle Auction designed by Wurman and Wellman [30], is a progressive anonymous-price combinatorial auction. It is an iterative auction, in which new bids are submitted in each iteration.

2.3.1 Model

Each agent submits bids for the bundles (combination of items) he is interested in. The auctioneer computes the temporary allocation and the price lattice to be revealed to the agents. The details about computing price is explained below. The agents can then increase their bids on the current bundles, or bid higher on some other bundles in the next iteration. The agents' bids always have to increase or stay the same with each bid submitted, but cannot decrease. The auction terminates when no agent wants to bid any higher, and all agents are satisfied with the current solution.

2.3.2 Equilibrium Bundle Prices

Once the allocation is determined, the prices need to be computed for the allocated bundles. An agent's *surplus* or *utility* for a bundle is the difference between the agent's value for the bundle and the price the agent would pay. Each agent that has been allocated a bundle has a surplus greater than or equal to 0, and 0 if not allocated a bundle. Also, a price for a bundle is at least as great as the price of its subsets. Prices are anonymous which means that the price lattice is the same for all agents.

Consider the example shown in Table 2.1. As we know, the optimal allocation is to give bundles AC and B to Agents 2 and 3, respectively.

Wurman and Wellman [30] have proven that the prices in AkBA are always in bundle-price equilibrium, when agents have values that are monotone on the finite lattice B. The following 3 steps are performed in order to compute the prices:

1. Solve the winner determination problem to get the socially efficient solution f^* whose value is $\mathrm{V}(f^*)$.

2. Since, f^* is the efficient solution, let I_θ be the set of agents who are allocated items in f^* and $I_{\neg\theta}$ be the agents who are not allocated a bundle. Let the bundles in f^* be denoted by B_θ and $B_\emptyset$ be the unassigned bundles. For each agent $i \in I_{\neg\theta}$, let ϕ_i be a null item which represents the agent's null allocation, and $\Phi \equiv \{\phi_i | i \in I_{\neg\theta}\}$. Let $G = B_\theta \cup \Phi$, and $g \in G$. G and the corresponding agent valuations describe an assignment problem.

   To compute the prices for all g, we need to compute the solution to the subproblem. This is solved using the dual program [12], by solving linear program LP_{lower} and LP_{upper}. The subproblem for our example in Table 2.1 is shown in Table 2.3.

   Let s_i be the surplus achieved by agent i. LP_{lower} maximizes each agent's surplus and is used to compute the lower bound prices in equilibrium. It can be represented by the following linear program,

$$\min \quad \sum_g \pi_g$$

Agent	AC	B	$\emptyset$
1	5	5	0
2	9^*	1	0
3	7	10^*	0

Table 2.3: Assignment subproblem to compute prices

$$\begin{aligned} \text{s.t} \quad & s_i + \pi_g \geq v_i(g), \forall i, g, \\ & s_i, \pi_g \geq 0, \\ & \sum_i s_i + \sum_g \pi_g = V(f^*) \end{aligned}$$

Similarly, LP_{upper} computes the upper bound prices,

$$\begin{aligned} \min \quad & \sum_i s_i \\ \text{s.t} \quad & s_i + \pi_g \geq v_i(g), \forall i, g, \\ & s_i, \pi_g \geq 0 \\ & \sum_i s_i + \sum_g \pi_g = V(f^*) \end{aligned}$$

3. After computing the solution to either LP_{lower} or LP_{upper} the prices for all $b \in B_\emptyset$ can be computed using the surpluses computed in step 2 as,

$$\pi_b = \max_i [v_i(b) - s_i] \tag{2.1}$$

Table 2.4 shows the lower and upper bound prices computed for the example in Table 2.1 using the above linear program.

2.3.3 Some Properties

Wurman and Wellman state and prove some desirable properties for the upper and lower bound prices. We list them in this section, for proofs and other details

Prices	A	B	C	AB	AC	BC	ABC
$\underline{\pi}^*$	0	3	3	10	9	13	18
$\overline{\pi}^*$	2	6	8	10	9	13	18

Table 2.4: Equilibrium Price Lattices

refer to [30].

Let $\underline{\pi}^*$ be the lower bound price obtained by LP_{lower} and $\overline{\pi}^*$ be the upper bound price obtained by LP_{upper}.

- The bundle prices, $\underline{\pi}^*$, support the efficient allocation.
- The bundle prices, $\overline{\pi}^*$, support the efficient allocation.
- For all k $\in$ (0,1), $k\overline{\pi}^* + (1-k)\underline{\pi}^*$, is in non-linear anonymous price equilibrium.

If k=1, only the upper bound prices are considered. This is a special case of the ascending k-bundle auction known as A1BA. We use the A1BA in our experiments.

AkBA maintains anonymous prices and the prices are computed for each bundle; the agent who wins a bundle pays the price announced for that bundle. Prices in AkBA are always in equilibrium with respect to the bids.

2.4 Bidding Strategies in AkBA

A strategy S is a sequence of bids that the agent submits to the auctioneer. The agent selects a strategy based on the agent's preferences, its estimation about the other agent's preferences, and the intermediate information revealed during the auction.

For GVA, it is a weakly dominant strategy for each agent to bid it's true valuation.

In ascending auctions, selecting the best strategy becomes more complicated, since intermediate information is revealed and a decision has to be made in every round as to what to bid next. We discuss a simple strategy called *myopic bidding* with respect to the AkBA.

Agent	A	B	AB
1	6*	3	9
2	5	4*	9

Table 2.5: Example to show myopic bidding

2.4.1 Myopic Bidding

Myopic bidding is a straightforward, best-response strategy for progressive auctions. In each round the agent bids on the bundle that gives it the highest surplus as if this were the last round of the auction.

In AkBA, at the end of each round the auctioneer tells each agent what they are currently winning and announces the highest bids for the bundles. A myopic agent behaves as if it can win any bundle that it is currently not winning by bidding δ more than the announced price, π_b, for the bundle b. If $\breve{b}$ is agent i's tentative allocation as announced by the auctioneer, then the myopic agent's strategy is to bid on the bundle, b', that maximizes its real surplus at the given prices,where,

$$b' = \arg\max_b \begin{cases} v_i(\breve{b}) - \pi_{\breve{b}} & \text{if } b = \breve{b}, \\ v_i(b) - (\pi_b + \delta) & \text{otherwise.} \end{cases} \tag{2.2}$$

If the above solution gives strictly more surplus than the agent's current allocation, the agent will increase its bid on b' to $\pi_{b'} + \delta$.

Consider the 2 agent, 2 item problem in Table 2.5. The agent's myopic strategies for each round are shown in Figure 2.1. The x-axis denotes the bundles, and y-axis denotes bid amounts. Agent 1's bidding behavior is denoted by solid lines, and agent 2's bidding by dotted lines. The numbers next to the bid values denote the round at which the bid was placed.

Wurman and Wellman[30] have previously studied the performance of A1BA (AkBA with k=1, refer to Section 2.3.3) on randomly generated problems when agents follow the simple myopic bidding strategy. However, in reality, agents are less likely to bid myopically if they can identify a strategy that yields a better surplus.

2.5 P-A1BA

An agent can use a *proxy agent* [22] to bid on his behalf. A proxy agent bids on behalf of the agent, based on the information that the agent provides about his valuations. The value submitted to this proxy agent is a *proxy value*, which means that the proxy agent bids the minimum on behalf of the agent until it reaches the proxy value.

We denote a proxy version of the A1BA auction as P-AkBA and illustrate it in Figure 2.2. When using a proxy agent, a strategy, S, is a sequence of proxy vectors $\{\alpha_1, \alpha_2, ...\}$ to submit to the proxy agent, where, $\alpha_k > \alpha_{k-1}$. Each α is a vector of values for the bundles.

The strategy is composed of several steps. Each step is a list of proxy values for all the bundles for that agent. Each agent starts by submitting its first vector of values for the bundles to the proxy agent. The proxy agent plays the auction and returns with the allocation and prices. At this point, the agent decides if it will move to the next step. The next step is another vector of values, but it is required to be an increment over the previous vector. When an agent moves to the next step, the new proxy values are computed and the auction continues with the new values.

2.5.1 Space of Strategies

To explore strategic behavior in A1BA, we need a good search algorithm that will search through the space of strategies to find a reasonably good strategy. The space of strategies is multi-dimensional and extremely large. Even for two agent, two-item problems, the space is six dimensional, one dimension for each agent-bundle combination. In our initial investigations, we have observed that the topology of the space is not uniform and it is difficult to solve analytically. Consider a simple 2 agent, 2 item abstract problem. We represent a possible slice of the landscape for the problem as shown in Figure 2.3. Agent 1 is the column player, and agent 2 is the row player, and we vary the values for, say, bundle A for Agent 1 and bundle B for Agent 2, keeping values for all other bundles at their true valuations. Each cell

is a joint strategy, and an agent can move unilaterally to a strategy (cell) that gives it better surplus. We have observed several patterns in the landscape which give us insight on why the search becomes difficult sometimes. In Figure 2.3 (a), there is a slope to the landscape. Agents move down towards their best strategy and hence can find a strategy that yields better surplus to both of them. In (b), the best strategy is in the middle of the loop, and agents cannot reach it unless they start at that strategy because there is no unilateral path to move to that strategy. In (c), there is no pure strategy equilibria for the agents to find, thus they get stuck in a loop. We have observed that such hills and ridges exist in the landscape of A1BAs, and make the search difficult.

We employ a *Genetic Algorithm* as a stochastic search method to search for better strategies. The following chapter discusses the reasons for using a genetic algorithm in this setting and describes our implementation.

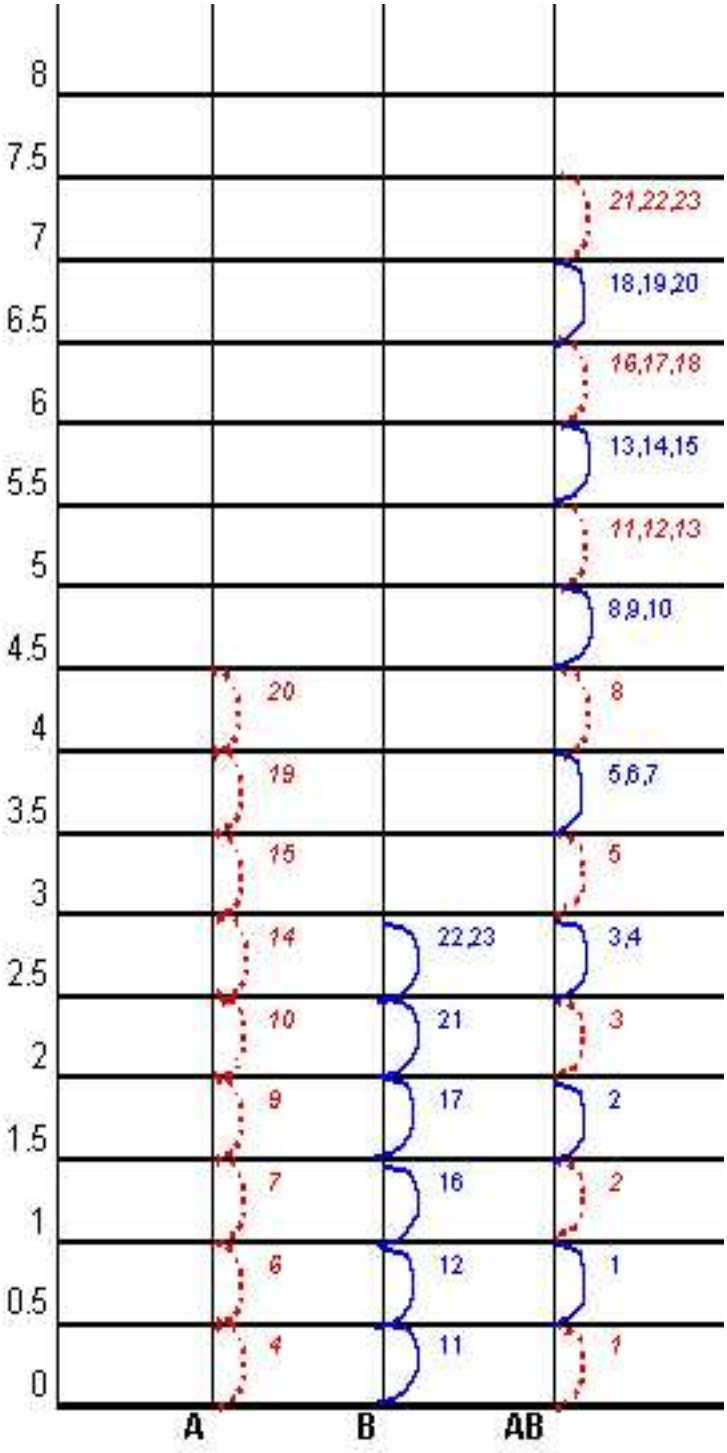


Figure 2.1: Myopic bidding for problem in Table 2.5

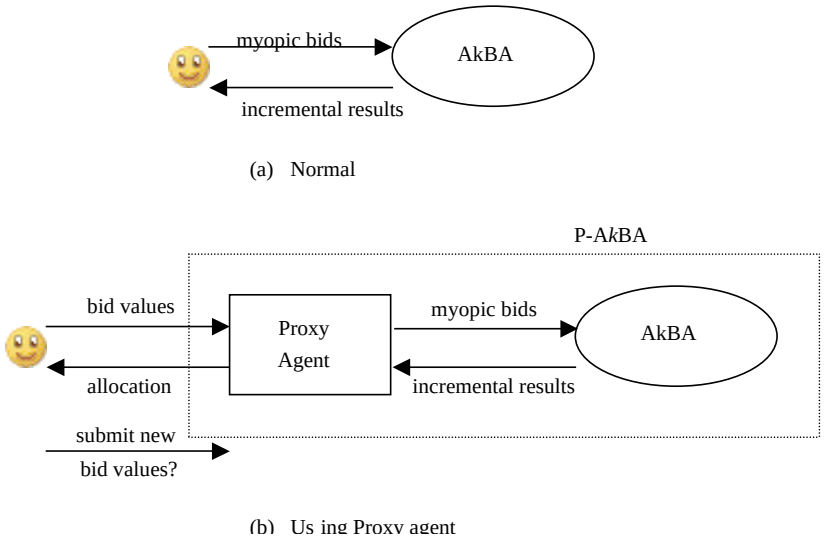


Figure 2.2: Proxy Bidding in P-A1BA

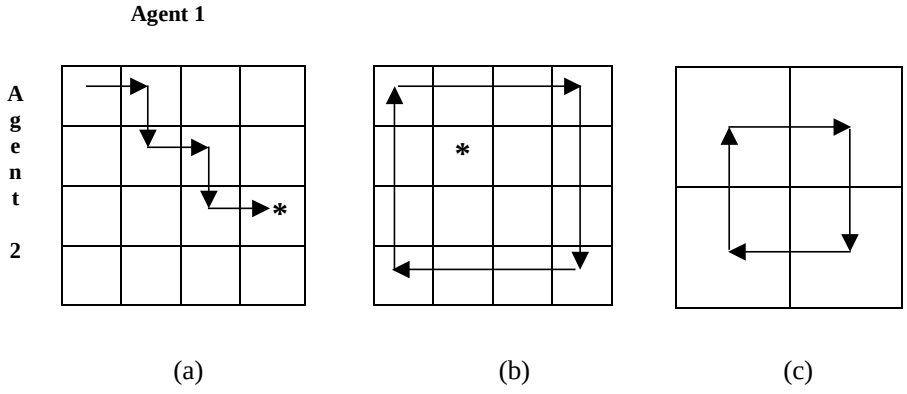


Figure 2.3: Slices of sample landscapes

Chapter 3

Evolutionary Computation

Genetic Algorithms (GAs) are a common, evolutionary optimization approach, best suited for large problems where locating the globally optimal solution is difficult [18]. GAs are particularly applicable to problems that are large, non-linear and possibly discrete in nature, features that traditionally add to the degree of complexity. GAs have been successfully used in complex, noncooperative settings to learn strategies for classic auctions [2], or repeated prisoner's dilemma games [3].

Genetic Algorithms (GAs) were invented by John Holland in 1970s with the idea of studying natural adaptations and to develop ways in which these adaptive processes can be applied to computer systems. Since then, GAs have evolved to have much wider applications, including economic models.[1]

3.1 Genetic Algorithms: Terminology

In a genetic algorithm, candidate solutions to a problem are encoded as *chromosomes*. A chromosome is conceptually divided into *genes*, each of which encodes a particular protein. A gene can be thought of as encoding a trait [18], and it has a fixed location in the chromosome.

The GA begins with a randomly created population of chromosomes (individuals). Each individual always represents a complete solution to the problem we are trying

[1]For an overview of different applications see [9], pages 126-129.

to optimize. Typically each individual is evaluated and the *fitness* of the individual is computed. The fitness of an individual is a function that determines how "good" an individual is. The GA creates a new population by selecting the best individuals (based on the fitness function) and allowing them to survive in the next generation. These candidate solutions or individuals are essentially bred with each other for several simulated generations and the principle of survival of the fittest is applied. In a computational setting, this means that the probability that an individual solution will pass on some of its parameter values to subsequent children is directly related to its fitness.

Crossover and *mutation* are GA operators used to modify the individuals and produce new offsprings. Crossover, as the name suggests, consists of exchanging parts of two individuals (parents) to produce new offsprings. The main objective of the crossover operator is to get genetic material from the previous generation to the subsequent generation and to explore different possibilities.

Mutation is used to introduce a certain amount of randomness to the search. It changes the value of a randomly chosen gene on an individual. It sometimes helps the search to find solutions that crossover alone might not encounter.

The individuals are ranked based on the output of the fitness function. Then, only some portion of the population is selected to reproduce.

3.2 A Framework

A simple genetic algorithm is shown in Figure 3.1. The individuals go through the evolution process until the termination criteria is reached. The termination criteria can be a fixed number of generations, or some condition such as a lower limit on the value of the fitness function.

A genetic algorithm has the following components:

- Encoding technique: The individual is usually represented in binary, and the string can be encoded by a function suited to the application of the GA.
- Initialization procedure: Creation of the population.

1. L =set of l individuals
2. Evaluate population by computing $f(l)$
3. Sort L by $f(l)$
4. While ($TerminationCriteria = false$)
   Select top m members of L'← L
      Create new members by *crossover*, add to L'
      Mutate
   L ← L'
   Evaluate population by computing $f(l)$

Figure 3.1: Genetic Algorithm

- Evaluation function: The fitness function is suited to the environment in which the GA is to be used. This function has to be defined carefully, as it can affect the search for a solution to a great extent.
- Selection and reproduction: This can be varied by changing the crossover criteria and the mutation rate.

The parameters in a GA can be modified by practice and art to improve performance. There is no rigid rule to set the parameters, or to evaluate their performance. The fitness of individuals can be mapped into a *landscape* where topography of this landscape varies according to the fitness function. A lot of work has been done in this area, for details refer to [7, 9, 18, 20].

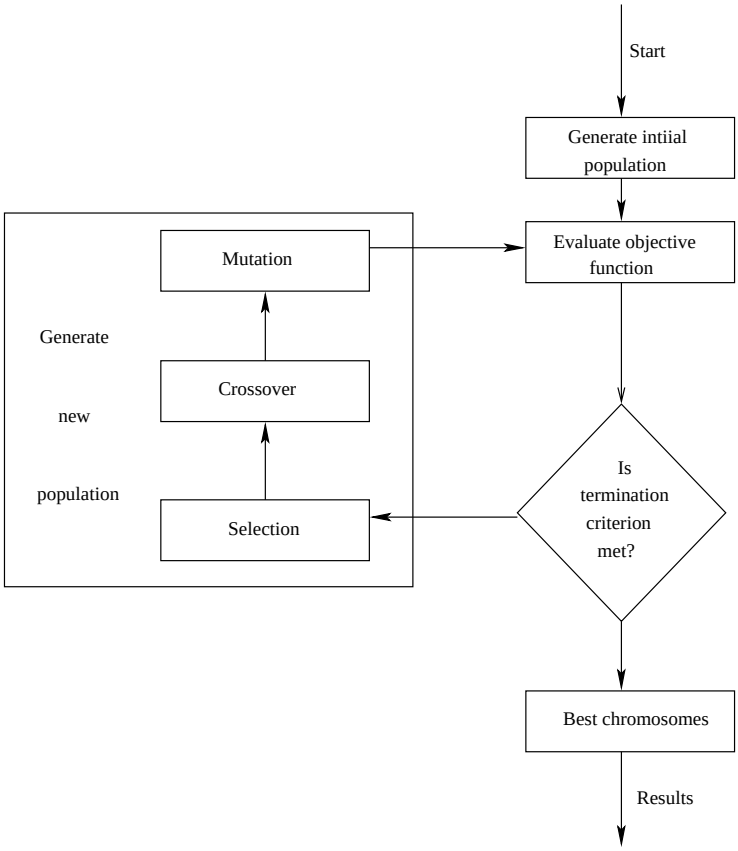


Figure 3.2: A Generic GA

Chapter 4

Genetic Evolution of Strategies

We apply a simple GA to search the combinatorial bidding space to find a reasonably stable strategy for an agent. The algorithm is run as a simulation of a mental process in which the agent is looking for a good strategy to play in the real auction. We use the A1BA in our experiments.

The simulation makes the following assumptions in the search:

1. The number of agents participating in the auction is fixed.
2. The agent has information about its preferences.
3. The agent has knowledge about the other agent's preferences.
4. The agent considers the possible strategies of each participant in turn.

A *proxy agent* [22] bids on behalf of each agent, based on the information that the agent provides about his valuations. The value submitted to this proxy agent is a proxy value, which means that the proxy agent bids the minimum required on behalf of the agent until it reaches the proxy value. We search the space of values that can be submitted to the proxy agent. This proxy agent is independent of the GA, and is only concerned about taking a vector of values from an agent, using these values to participate in the auction, and returning the outcome of the auction to the agent. The agent then decides if he wants to continue by bidding further, or to stop at that

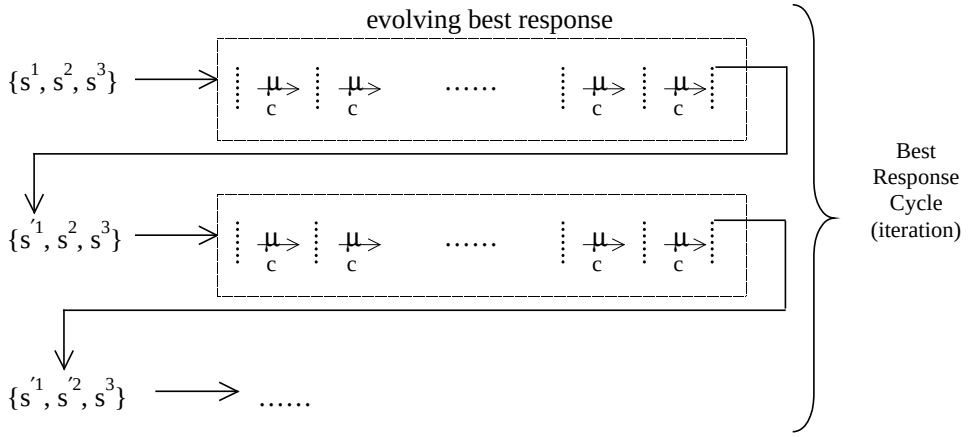


Figure 4.1: Best Response Strategy

point. If the agent wants to continue, it submits the next vector to the proxy agent and the auction continues.

4.1 Algorithm

Each agent $i \in I$ has a population of individuals, denoted by L_i. An individual $l \in L_i$ is a random string of 0s and 1s that we translate into the agent's strategy s. The mapping from l to s is described in Section 4.3. The number of individuals in a population is pre-determined by the population size p. Each strategy s is made up of a number of vectors we call *steps*, that is $s = \{\alpha_1, \alpha_2, ...\}$

4.1.1 Best Response Search

We assume the agent considers the possible strategies of each participant in turn. During its turn, agent i tries all of its individual strategies against the best strategies

of the other agents. After playing all the strategies with the best of the other agents', which we call a generation, the population is sorted so that the first member is the best individual. Figure 4.1 shows the *best response cycle*. The symbols 'μ' and 'c' denote mutation and crossover explained in Section 4.2.1. At the end of every agent's turn, the set of initial strategies is updated with that agent's best evolved strategy.

The process continues for a number of generations, g, with each agent taking turns selecting a strategy trying to improve its outcome.

4.2 Modules

The GA is implemented to interact with the auction in stages, and the algorithm is interfaced with the auction as shown in Figure 4.2. The "GA Module" is responsible for creating populations of individuals for each agent, applying mutation and crossover to the population and repeating the process for a specified number of iterations.

The "Auction Module" takes one individual for each agent, as the agent's strategy from the GA-Module and interprets it as the agent's values for the bundles as explained in Section 4.3. It runs the A1BA using these values and computes the utility for each agent. The utility is the fitness of the individual and it gets associated with the individual. After all individuals are analyzed to compute the fitness, the population is sorted based on the fitness and returned to the GA-Module. GA-Module then applies crossover and mutation, and generates a new population. This process continues iteratively until none of the agents are willing to change their strategy or until it reaches a limit on the number of iterations, whichever comes first. When the process terminates, every agent's best individual so far is picked as the strategy for each agent. The k-bundle auction is run again for these values to verify the results.

4.2.1 Selection

Only the best few individuals are selected to move to the next generation. A new population is created by doing a crossover. Two individuals are randomly picked from the top one-third of the current sorted population. The crossover function splits both

these individuals at a random point and rebuilds a new individual. Once the new population is created, it undergoes *mutation*, at a pre-determined mutation rate μ.

4.2.2 Step Advancement

As we discussed earlier, each individual is composed of several steps. Each step has a list of proxy values that are an increment over the previous step. The agents start by submitting their first step as their proxy valuations to the proxy agent. When the proxy agent returns the result of the auction with those values, the agent decides if it needs to submit the next step or stop at that point.

The primary goal of each agent is to maximize its utility, and hence an agent will move to the next step only if the utility that the agent will get if it bid in the next step is higher than the current utility of this step. If this does not hold true then the agent is happy with the current utility, and hence the current utility is recorded as the fitness of the individual.

4.3 Encoding Scheme

The individual is interpreted as a sequence of steps, σ, where each determines the fraction of the true value submitted as the *proxy value* to the auction. We compute this proxy value as the increment over the previous step's proxy value for each bundle. The resolution, r, decides the number of bits to use to determine the fraction of the true value. The length of the individual, ℓ, is a function of the resolution, r, number of steps, σ, and the number of items, m.

$$\ell(r, \sigma, m) = r\sigma(2^m - 1)$$

Consider an agent i with $\sigma = 3$, $r = 2$, $m = 2$. To interpret the individual 101110001101010101 we first break it into three steps. Hence, we have,

$101110001101010101 \Rightarrow [10, 11, 10][00, 11, 01][01, 01, 01]$

Then, we divide each step into 3 words of two bits each, where the word corresponds to a multiplier or a fraction for a particular bundle.

$[10, 11, 10][00, 11, 01][01, 01, 01] \Rightarrow [2/3, 1, 2/3][0, 1, 1/3][1/3, 1/3, 1/3]$

For a step $x < \sigma$ for agent i, the fraction generated is $f_i^x(b) <= 1$. Agent i's true value for bundle b is $V_i(b)$. The proxy value for the bundle b would be,

$$V_i^x(b) = [1 - \prod_{x=1}^{\sigma}[1 - f_i^x(b)]]V_i(b)$$

If $V_i =< 3, 5, 7 >$ is the true valuation vector for agent i, then the proxy values computed for all the three steps would be $[2, 5, 14/3][2, 5, 49/9][7/3, 5, 161/27]$. Each agent submits a proxy value for each bundle computed using the above rule. Thus, for the individual $l = 101110001101010101$ in this example the strategy would be $s = [2, 5, 14/3][2, 5, 49/9][7/3, 5, 161/27]$

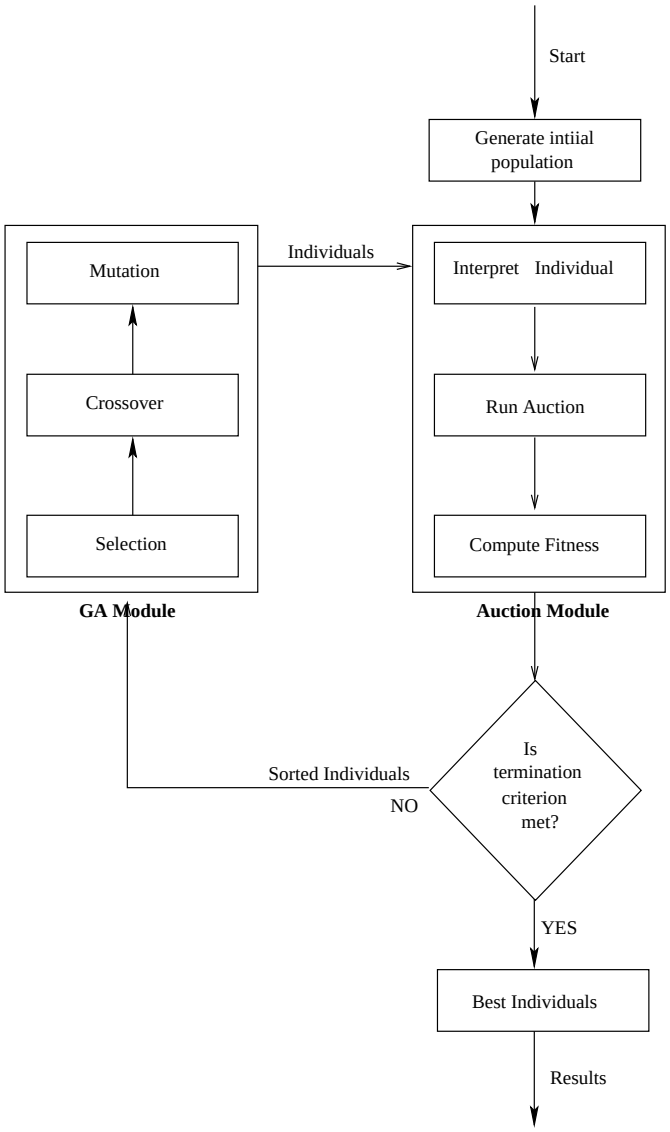


Figure 4.2: Process of evolving a best response

1. Repeat for k iterations
   2. Make population for all agents
   3. Repeat for each Agent $i \in I$
      Repeat for each generation
         For each $s \in S_i$
            Compute fitness by solving AkBA $(s, s^{*j \neq i})$
         Make new Population by Crossover
         Mutate
      If (*current-fitness* > *original-fitness*)
         Agent i moves his strategy
         Set new strategies for agent.

Figure 4.3: Iterative Tournament

1. For each member $s \in S_i$ of the population do the following
   2. Get each agent i's valuations from his individual
   3. *initial-bids* $\leftarrow 0$; *current-step-list* $\leftarrow 0$; *unhappy* $\leftarrow$ true
   4. while (*unhappy* $\leftarrow$ true)
      Run k-bundle auction with new initial bids
      decide next-step for agent $i \in I$
      if agent i is not moving to next step
         unhappy $\leftarrow$ false
      fitness(s) $\leftarrow$ Utility of Individual σ
      Get each agent i's valuations from his individual

Sort individuals in the population by fitness

Figure 4.4: One Iterative Generation

Chapter 5

Experimental Analysis

We have implemented the GA to interact with A1BA. The entire model is coded using LISP.

5.1 Design of Experiments

5.1.1 Hypotheses

We have several hypotheses which we wanted to test in our experiments.

- If agents cooperate with each other then they will not need to move to the next step; step 2 (or later) is useful primarily as a punishment stage.
- The agent finds a strategy that yields better surplus, than myopic bidding.
- The search finds solutions that are in Nash Equilibrium.
- The Nash equilibria of the game might prevent the agents from evolving to a strategy that gives the optimal solution.

In addition to these hypotheses, we also wanted to investigate:

- The ability of the search to find a good solution given a set of parameters.
- The sensitivity of the search to changes in the parameters.

5.2 Parameters

The search quality depends on the GA. We explored the GA parameters to determine a fixed set before running the experiments.

We conducted experiments for two broad categories of problems as explained in Section 5.2.1 below. When testing for the performance of the search, all parameters were at the default values specified. Parameters were tweaked only when testing for the sensitivity.

5.2.1 General Parameters

1. Problem Category: We have run experiments for randomly generated problems. We created problems for 3 categories:

   - Complementary preferences: Randomly generated problems where all agents exhibit complementarities.
   - Substitutable preferences: Randomly generated problems where all agents exhibit substitutablee preferences.
   - General preferences: Randomly generated problems with any kind of preference. This was the *default* category.

2. Number of agents: Problems were run with 2 or more agents.

3. Number of items: All problems were run with 2 or more items.

4. Bid increment: This was set to a constant 0.5.

5. Sensitivity to starts: For testing sensitivity to the starting point, the same problem with the same set of parameters was run for a number of times to check if the starting point in space made a difference to the search.

6. Number of iterations: This was set by trial and error, to 10. Also, the search terminated if none of the agents changed their strategy for two consecutive iterations.

The defaults were 3 agent, 3 item (7 bundles) problems with general preferences and a bid increment of 0.5.

5.2.2 GA Parameters

A major decision that had to be made before running the experiments was setting the parameter values for the GA, including the population size, p, mutation rate, μ, number of generations, g, and the encoding parameters like the resolution factor, r, and the number of steps, σ. Changing these parameters greatly affected the performance of the search, and hence a considerable amount of time was spent before we settled on a set of parameters.

The population size and the number of generations were the most difficult to settle on, since they affected the search in both its ability to find good solutions and the amount of time spent. Moreover, they are interdependent and cannot be optimized one at a time. With a large p and small g, the search could not fully explore the space. Reducing the population size and number of generations resulted in premature convergence. Having large values for both these parameters resulted in a very high run time. Finding appropriate values for these parameters required experimentation. By trial-and-error we fixed the following values:

1. Population size, p: This parameter was varied when testing for sensitivity from 25-100.

2. Number of Generations, g: We varied the number of generations and found that 25 was a sufficient value for our problem size.

3. Mutation Rate: This is varied from 0.05 to 0.5.

4. Resolution, r: This has been varied among 2, 3, 4 and 5 bits. We selected 4 as the default value because 3 seemed not enough resolution, and 5 gave 2^5 possible fractions which were more fine grained than the bid increment.

5. Number of Steps, σ: Values of both 2 and 3 were explored. It was observed that 2 steps were sufficient.

The default values were set as $p = 50, g = 25, \mu = 0.05, r = 4, \sigma = 2$.

With $r = 4$, we have 2^4 possible fraction values for each bundle, and with 3 items the 7 bundles would give a total of 16^7 possible strategies. With $p = 50$, we are searching with a population of 50 individuals in a space of 16^7 possibilities. With 2 steps, there are 16^{14} possible strategies.

Even for small problems, the computation is quite complex. Every agent tries all his individuals with the current best responses of the other agents. Since we run our experiments for 3 agents, $p = 50$ and $g = 25$, one evolutionary cycle requires $3 \times 50 \times 25 = 3750$ A1BAs be run. We repeat this process with new populations for 10 cycles, and could run as many as 37,500 auctions. In our experiments, the minimum number of auctions run is 11,250 – a case where no agents changed strategies for 2 consecutive cycles.

5.2.3 Data Recorded

All problems were created randomly within the constraints of the problem group. We recorded the following data about the results of the search:

1. Original problem
2. Original Problem Solution
    - Allocation
    - Utility
    - Surplus
    - A1BA prices with myopic bidding
3. Evolved Solution
    - Evolved strategies
    - Allocation
    - Utility

- Surplus
- P-A1BA prices with evolved bidding
- Step number of each agent
- Iteration number at which search terminated.

4. Sealed-bid Prices for Original Problem (upper and lower bound)
5. GVA Payments for Original Problem

Chapter 6

Results and Conclusions

We ran a total of 60 distinct problems; 20 for each of the categories mentioned in Section 5.1. We did not observe any distinctive behavior within the different categories. Hence, all the observations we make about the search, game equilibrium, and auction results are for the set of 60 problems in general.

For testing sensitivity to the parameters mentioned in Section 5.2, we chose 3 problems and ran them 10 times each to make observations. These problems did not show any markedly different results in the search or auction solution when run several times with different values of mutation rate and resolution.

We looked for insights about the solution, equilibrium, *Pareto dominance*, and collaborating agents. A solution is Pareto efficient if there is no other solution in which some other agent is better off and no agent is worse off. If in solution x someone is better off and no one is worse off than in the solution y, we say that x Pareto dominates y. In our case, we compare the surplus of all the agents achieved by myopic bidding against the surplus achieved by playing the evolved strategies. If the evolved strategies yield (weakly) better surplus for all the agents, and strictly better for one, we say that the evolved solution Pareto dominates the myopic strategy. We show our results in Figure 6.1. Out of the 60 problems, in 14 problems the agent found a set of strategies that gave the optimal solution. For all these problems the surplus of each agent was at least as much as the surplus of the myopic strategies. For the 46 other problems, we checked for properties of Pareto dominance and total

<table>
<tr><td rowspan="3">Problem set
60</td><td>Optimal allocation
14</td><td colspan="2">$\forall$ i, surplus (evolved) > surplus (original)
14</td></tr>
<tr><td rowspan="2">Sub optimal allocation
46</td><td>Pareto dominance</td><td>15</td></tr>
<tr><td>Higher Σ surplus</td><td>33 (18 more than Pareto)</td></tr>
</table>

Figure 6.1: Results

surplus comparison. 33 of the 46 problems gave a total surplus greater than or equal to the sum of the surpluses of all agents with myopic bidding, though this increase in overall surplus came at the expense of one of the agents. 15 out of 33 problems gave solutions that were Pareto dominant, which means, for all these problems, each agent achieved a higher than or at least as much surplus with the evolved strategy as with myopic bidding.

In only 2 problems, did an agent evolve a strategy where the agent submitted another proxy vector as step 2. In all of the other problems all agents stopped at step 1 of the strategy. This could be because agents cooperate, since step 2 could be a threat to an agent that deviates from the collaborative solution. In one of two cases, the agent that used 2 steps of the strategy won the bundle ABC in the solution even though the optimal allocation assigned it only A. This can be possible because an agent that dominates the auction is indifferent between strategies that achieved dominance in one step or two. This example does not support the hypothesis about the punishment phase yet. Further investigation is required to answer the question of the value of the second step, but the evidence collected so far does not disprove it.

For the problems in which the evolved strategies gave the optimal allocation, the utility was at least as much as the utility in the original problem solution. Figure 6.2

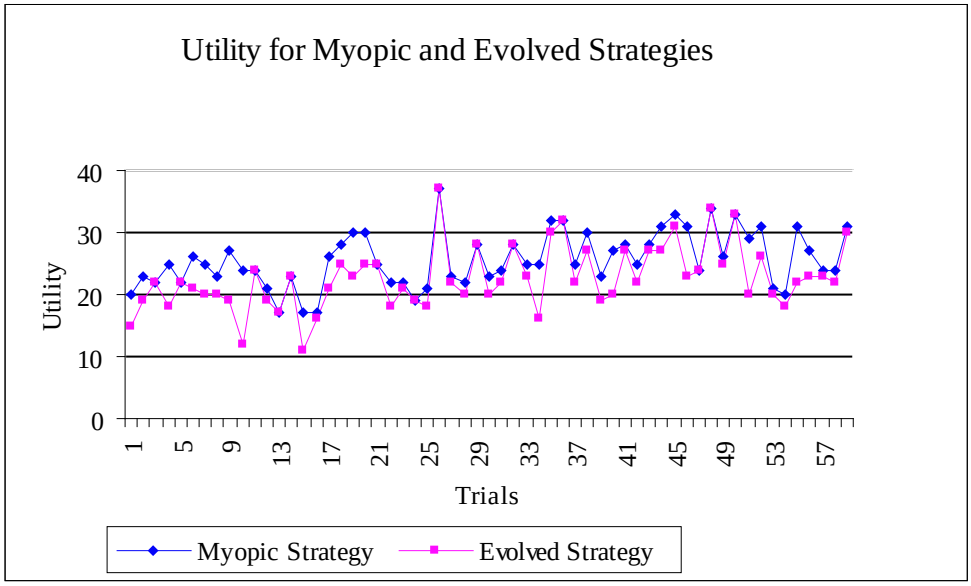


Figure 6.2: Utility comparison for Myopic vs Evolved Strategies

illustrates the utility comparison for the myopic and evolved strategies. About 25% of the problems converged to a strategy that gave the optimal solution. For these problems, the agents with the evolved strategies achieved at least as much and often higher surpluses than the myopic strategies. Figure 6.3 shows the surplus comparison for problems that (a) found optimal solutions (b) found sub-optimal solutions.

We observed that the same problem run with the same set of parameters often gives different results. This is possible due to the complex nature of the strategy space, the existence of multiple game equilibria, and the stochastic features of the GA. Also, due to the iterative best-response search, the agents can only move to solutions they can reach by unilateral improvements, (Figure 2.3 shows cases where it does not work). Hence, the agents sometimes cannot find a path to move to the optimal solution.

To show that P-AkBA has topological features like those in Figure 2.3, we ran the problem shown in Table 6.1 three times. The optimal solution is to give A to Agent 1 and BC to Agent 2. Three runs of the search found 3 sets of strategies for agents 1, 2

Agent	A	B	C	AB	AC	BC	ABC
1	10*	4	12	8	13	8	15
2	6	10	17	10	11	26*	34
3	4	7	9	10	12	16	19

Table 6.1: Strategy Game Problem

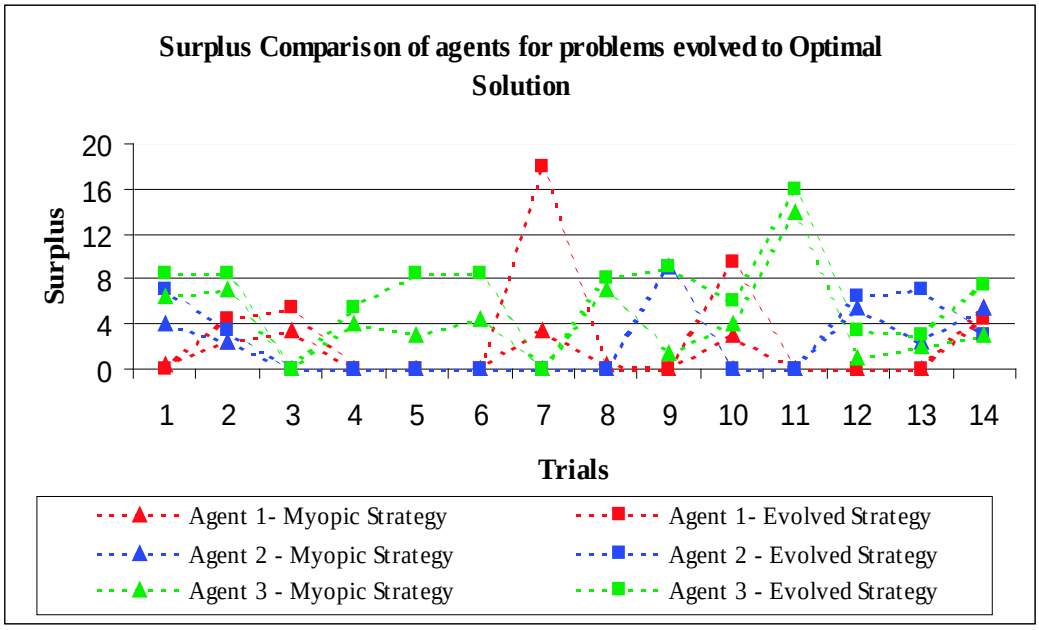
and 3, denoted by $\{s_1^1, s_1^2, s_1^3\}, \{s_2^1, s_2^2, s_2^3\}, \{s_3^1, s_3^2, s_3^3\}$. These 3 sets of joint strategies represent two different solutions. We assume, for the purpose of illustration, that since Agent 3 has low valuations and was not a part of the solution, it does not have enough power to affect the solution. Hence, for simplicity, we fix s_1^3 as Agent 3's strategy for the example. We ran the k-bundle auction using the strategies for the other two agents and completed the table shown in Figure 6.4. In this figure, the rows are strategies for Agent 2 and the columns indicate strategies for Agent 1. The top right corner in each cell represents the allocation found by the joint strategies.

Now consider Agent 1 when Agent 2 is playing s_1^2. Agent 1 is indifferent between any of its 3 strategies because it gets a surplus of 0. If it chooses s_1^1, Agent 2 wins all three items, which is a non-optimal solution. In this abstracted subset of the game, this solution is a Nash equilibrium. Neither agent can make itself better off by unilaterally deviating from that strategy. On the other hand, if Agent 1 chooses s_2^1 as its strategy, then Agent 2 plays s_2^2 and again wins ABC. However, if Agent 1 plays s_3^1, then Agent 2 is able to play s_3^2, which leads to the optimal allocation. These observations illustrate the fact that there are large regions of indifference in the strategies, and the agents' selection of a strategy in that region affects the search to a great extent.

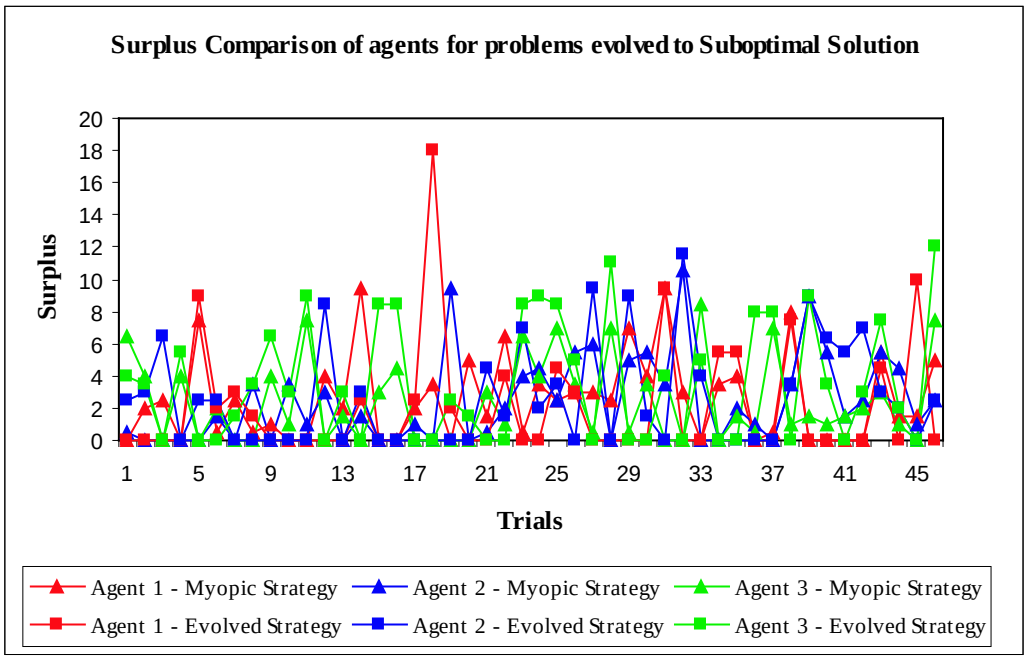
We also compared prices for the evolved strategies with the GVA payments for the myopic strategies. For all the 14 problems that evolved to strategies that gave an optimal solution, the evolved prices were lower than, or equal to, the corresponding GVA payments for the myopic strategies. Since GVA payments are computed for winning agents and not for each bundle, a comparison cannot be made for the 46 problems that give a solution different from the optimal solution.

The agents achieved an increased surplus by finding strategies that result in lower

prices. We illustrate the relationship among different prices for the problem in Table 6.1 in Figure 6.5. We plot the prices for the winning bundles A and BC. The evolved strategies give lower prices than the prices for the myopic strategies. We also plot the sealed-bid low and high prices, $\underline{\pi}$ and $\overline{\pi}$. Also, the GVA payments were 8 and 16 for Agent 1 and Agent 2, respectively. As seen in the figure, the evolved prices are much lower than the GVA payments, and in fact dominates all other outcomes, from the point of view of the buyers.



(a)



(b)

Figure 6.3: Surplus comparison for agents. (a) Evolved strategies with Optimal Solution, and (b) Evolved Strategies with Sub optimal Solution.

Agent 2 \ Agent 1	S_1^1	S_2^1	S_3^1
S_1^2	(2,2,2) (19, 0)	(2,2,2) (12.7, 0)	(2,2,2) (14.2, 0)
S_2^2	(2,2,2) (9.9, 0)	(2,2,2) (16.5, 0)	(2,2,2) (7.4, 0)
S_3^2	(3,2,2) (11.5, 0) Agent 3 = 1.1	(1,2,2) (16, 5.8)	(1,2,2) (18.5, 6)

Figure 6.4: A slice of the three agent strategy space, when agent 3 is assumed to be playing s_1^3 for all the trials, and agents 1 and 2 play three different strategies against each other.

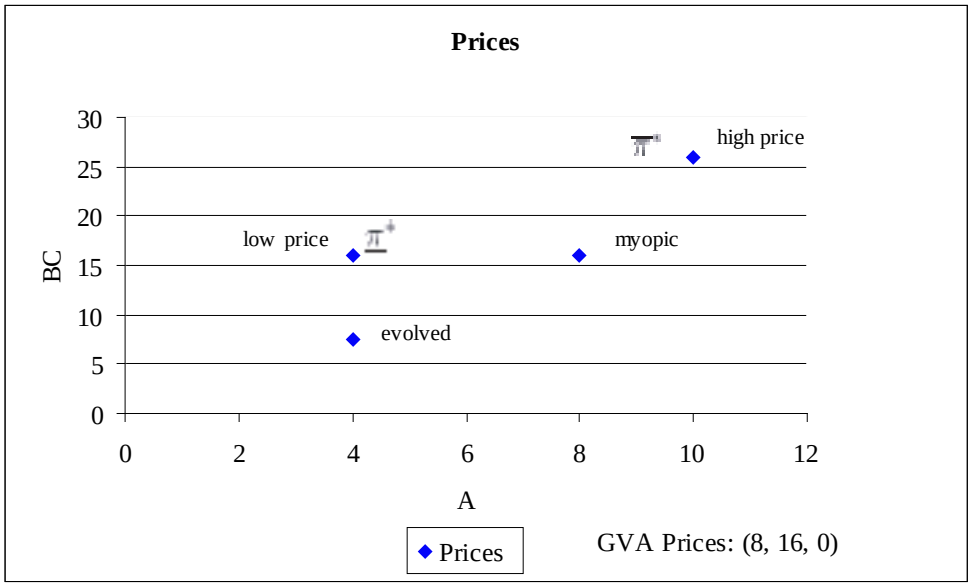


Figure 6.5: Prices for problem in Table 6.1.

Chapter 7

Related Research and Future Work

7.1 Related Research

There is a lot of ongoing research on Combinatorial Auctions, sealed-bid [14] as well as iterative [21, 23, 30]. The Winner Determination Problem has received a lot of attention [1, 25, 26]. Andersson *et al.* [1] have proposed to use a standard optimization software like CPLEX to solve the winner determination problem. Lehman, Callaghan and Shoham [11] discuss the effects of approximate winner determination in GVA. Nisan [19] provides an analysis of the computational problems in bidding languages in detail. Progressive combinatorial auctions seem to be more effective (or acceptable) in practical applications than the sealed-bid GVA. *i*Bundle [21], A*k*BA [30] and the Simultaneous ascending auctions(SAA) [16] are few of the recent auctions.

A lot of research has been done on GAs since their inception by Holland in 1970s. The fitness landscapes have been studied and an attempt to characterize them has been done by Smith [27]. Smith shows that a set of evolvability metrics can predict the difficulty of finding good solutions. Juille and Pollack [10] present a framework for coevolutionary learning with a population of learners exposed to search. In their framework, the training environment adapts in response to the progress of the learners to produce problems with increasing difficulty. For more information and references on work on GAs refer to [9, 18].

Outside of the simple classic auctions studied by economists, not much work has

been done in exploring strategic behavior for auctions. There have been some attempts to design sophisticated and efficient bidding strategies for agents participating in online auctions. GAs have been applied as a search tool for characterizing effective trading strategies for double auctions [17]. Miller compares 30 computer trading programs and analyzes the winning trading strategy. Andreoni and Miller [2] create and analyze a model of adaptive learning in double auctions. Their model can capture the bidding patterns using artificial adaptive agents and have shown that adaptive learning can provide some good insights into bidding behavior. Phelps *et al.* [24] apply a GA to evolve strategies for a double auction.

Neusite Solutions Limited, a UK based company, is the first known to develop a system that models the knowledge a bidder may have about his competitors in a combinatorial auction. It uses GAs and neural networks and is the only one to our knowledge to apply GAs to combinatorial auctions. However, the details of their technology are unavailable, and the company has now ceased operation.

7.2 Future Research Directions

We have explored the strategy space for only one type of CA, the A1BA. A comparison could be made by using a similar evolutionary approach for evolving bidding strategies for other iterative combinatorial auctions like the *i*Bundle [21]. The genetic operators used in our GA, no doubt were effective, but did not always guarantee an optimal solution. GA parameters could be adjusted on-the-fly [28], so that the agents keep evolving strategies until the utility is at least as much as that of the true solution. The results from the experiments can be used to provide insight on computing direct solution for the P-AkBA. More elaborate representations of strategies (other than using the P-AkBA) can be explored. Also, the evolved solutions are suspected to be Nash equilibrium, but it is yet to be proved that they always are.

Bibliography

[1] Arne Andersson, Mattias Tenhunen, and Fredrik Ygge. Integer programming for combinatorial auction winner determination. In *Fourth International Conference on Multiagent Systems*, pages 39–46, 2000.

[2] James Andreoni and John H. Miller. Auctions with artificial adaptive agents. *Games and Economic Behavior*, 10:39–64, 1995.

[3] Robert Axelrod. Evolution of strategies in the iterated prisoner's dilemma. In Lawrence Davis, editor, *Genetic Algorithms and Simulated Annealing*, pages 32–41. Morgan Kaufmann, San Mateo, CA, 1989.

[4] Carrie Beam, Arie Segev, and J. George Shanthikumar. Electronic negotiation through Internet-based auctions. *CITM Working Paper 96-WP-1019*, page 13, December 1996.

[5] James Case. Mathematical challenges of combinatorial auction design. *SIAM News*, pages 21–29, 2000.

[6] Ralph Jr. Cassady. *Auctions and Auctioneering*. University of California Press, 1967.

[7] Rajarshi Das, Melanie Mitchell, and James P. Crutchfield. A genetic algorithm that discovers particle-based computation in cellular automata. *Third Parallel Problem-Solving From Nature Conference*, March 1994.

[8] Sven de Vries and Rakesh Vohra. Combinatorial auctions: A survey. *INFORMS Journal of Computing*, 2002. forthcoming.

[9] David E.Goldberg. *Genetic Algorithms in search, optimization, and machine learning*. Addison-Wesley, 1989.

[10] Hugues Juille and Jordan B. Pollack. Coevolutionary learning: a case study. In *Proc. 15th International Conf. on Machine Learning*, pages 251–259. Morgan Kaufmann, San Francisco, CA, 1998.

[11] Daniel Lehmann, Liadan Ita O'Callaghan, and Yoav Shoham. Truth revelation in rapid, approximately efficient combinatorial auctions. In *ACM Conference on Electronic Commerce*, pages 96–102, 1999.

[12] Herman B. Leonard. Elicitation of honest preferences for the assignment of individuals to positions. *Journal of Political Economy*, 91(3):461–479, 1983.

[13] Kevin Leyton-Brown, Mark Pearson, and Yoav Shoham. Towards a universal test suite for combinatorial auction algorithms. In *Second ACM Conference on Electronic Commerce*, Minneapolis, 2000.

[14] Jeffrey K. MacKie-Mason and Hal R. Varian. Generalized Vickrey Auctions. Technical report, University of Michigan, July 1994.

[15] R. Preston McAffe and John McMillan. Analyzing the airwaves auction. *Journal of Economic Perspectives*, 10(1):159–175, 1996.

[16] Paul Milgrom. Putting auction theory to work: The simultaneous ascending auction. *Journal of Political Economy*, 108(2):245–272, 2000.

[17] John H. Miller, John Rust, and Richard Palmer. Characterizing effective trading strategies. *Journal of Economic Dynamics and Control*, 18:61–96, 1994.

[18] Melanie Mitchell. *An Introduction to Genetic Algorithms*, volume 34(5). MIT Press, 1998.

[19] Noam Nisan. Bidding and allocation in combinatorial auctions. In *Second ACM Conference on Electronic Commerce*, 2000.

[20] Jim R. Oliver. A machine learning approach to automated negotiation and prospects for electronic commerce. *http://opim.wharton.upenn.edu/oliver27/papers/jmis.ps: University of Pennsylvania*, page 7, July, 1996.

[21] David C. Parkes. *i*Bundle: An efficient ascending price bundle auction. *First ACM Conference on Electronic Commerce*, pages 148–157, November 1999.

[22] David C. Parkes and Lyle Ungar. Preventing strategic manipulation in iterative auctions: Proxy-agents and price-adjustment. In *17th National Conference on Artificial Intelligence, (AAAI-00), pp. 82-89*, 2000.

[23] David C. Parkes and Lyle H. Ungar. Iterative combinatorial auctions: Theory and practice. In *Seventeenth National Conference on Artificial Intelligence*, pages 74–81, 2000.

[24] Steve Phelps, Simon Parsons, Peter McBurney, and Elizabeth Sklar. Co-evolution of auction mechanisms and trading strategies: Towards a novel approach to microeconomic design. 2002.

[25] T. Sandholm, S. Suri, A. Gilpin, and D. Levine. Cabob: A fast optimal algorithm for combinatorial auctions. In *International Joint Conference on Artificial Intelligence*, Seattle, WA, 2001.

[26] Tuomas Sandholm. An algorithm for optimal winner determination in combinatorial auctions. In *Sixteenth International Joint Conference on Artificial Intelligence*, pages 542–547, 1999.

[27] Tom Smith, Phil Husbands, Paul Layzell, and Michael O'Shea. Fitness landscapes and evolvability. *Evolutionary Computation*, 10(1):1–34, 2002.

[28] M. Srinivas and L. M. Patnaik. Adaptive probabilities of crossover and mutation in genetic algorithms. *IEEE Trans. Systems Man Cybernet*, 24:656–667, 1994.

[29] William Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *Journal of Finance*, 16:8–37, 1961.

[30] Peter R. Wurman and Michael P. Wellman. AkBA: A progressive, anonymous-price combinatorial auction. In *Second ACM Conference on Electronic Commerce*, 2000.

[31] Peter R. Wurman, Michael P. Wellman, and William E. Walsh. The Michigan Internet AuctionBot: A configurable auction server for human and software agents. In *Second International Conference on Autonomous Agents*, pages 301–308, 1998.

[32] Peter R. Wurman, Michael P. Wellman, and William E. Walsh. A parametrization of the auction design space. *Games and Economic Behavior*, 35(1-2):304–338, Apr.-May 2001.

Appendix A

Sample Data

	Problem	Evolved		Myopic		Optimal	
		Allocation	Surplus	Allocation	Surplus	Allocation	Surplus
1	Agent 1:(0 2 7 11 5 8 13 18)		0		0		0
	Agent 2:(0 1 4 7 6 11 13 18)	BC	2.5	C	0.5	C	1
	Agent 3:(0 10 8 19 1 12 11 24)	A	4	AB	6.5	AB	8
2	Agent 1:(0 4 2 10 8 17 15 20)		0	AC	2	AC	6
	Agent 2:(0 1 4 7 9 11 14 19)	B	3		0		0
	Agent 3:(0 2 8 15 5 12 15 22)	AC	3.5	B	4	B	4
3	Agent 1:(0 9 10 24 6 19 21 32)		0	ABC	2.5	ABC	2
	Agent 2:(0 10 9 23 4 17 18 30)	ABC	6.5		0		0
	Agent 3:(0 1 10 15 2 6 13 18)		0		0		0
4	Agent 1:(0 6 7 16 7 18 15 28)		0		0		0
	Agent 2:(0 7 4 15 7 18 15 24)		0		0		0
	Agent 3:(0 10 8 23 6 18 17 32)	ABC	5.5	ABC	4	ABC	4
5	Agent 1:(0 10 7 19 2 14 13 25)	AB	9	ABC	7.5	ABC	7
	Agent 2:(0 4 4 12 3 12 9 18)	C	2.5		0		0
	Agent 3:(0 6 3 12 1 10 8 16)		0		0		0
6	Agent 1:(0 6 2 13 10 17 14 26)	AC	2			C	0
	Agent 2:(0 4 10 18 3 10 18 23)	B	2.5	C	1.5	B	0
	Agent 3:(0 10 7 18 8 19 16 28)		0	AB	2	A	5
7	Agent 1:(0 6 7 14 4 15 14 23)	C	3	ABC	2.5	A	2
	Agent 2:(0 3 8 15 1 6 12 17)		0		0		0
	Agent 3:(0 9 1 15 1 12 3 17)	AB	1.5		0	BC	5
8	Agent 1:(0 1 10 13 1 3 13 16)	BC	1.5	B	0.5	B	2
	Agent 2:(0 8 1 11 7 17 13 23)		0	AC	3.5	AC	5
	Agent 3:(0 7 8 19 2 12 15 23)	A	3.5		0		0
9	Agent 1:(0 6 5 14 6 17 15 24)		0	C	1	C	1
	Agent 2:(0 7 2 12 5 17 12 21)		0		0		0
	Agent 3:(0 10 9 22 2 13 12 27)	ABC	6.5	AB	4	AB	9
10	Agent 1:(0 3 1 5 7 15 9 18)		0		0		0
	Agent 2:(0 2 9 15 7 10 21 24)		0	BC	3.5	BC	5
	Agent 3:(0 4 7 14 6 12 17 22)	ABC	3	A	1	A	1

11	Agent 1:(0 5 4 10 6 13 12 19)		0		0		0
	Agent 2:(0 1 4 7 9 13 16 18)		0	C	1	C	3
	Agent 3:(0 9 7 19 6 19 14 27)	ABC	9	AB	7.5	AB	9
12	Agent 1:(0 8 9 21 1 10 13 25)		0	A	4	A	4
	Agent 2:(0 1 10 13 9 15 23 27)	ABC	8.5	BC	3	BC	8
	Agent 3:(0 4 4 10 8 15 15 21)		0		0		0
13	Agent 1:(0 6 10 19 8 15 23 32)		0	B	2	B	9
	Agent 2:(0 5 1 7 4 12 7 16)		0		0		0
	Agent 3:(0 10 4 15 10 23 18 31)	ABC	3	AC	1.5	AC	11
14	Agent 1:(0 3 10 17 6 14 21 25)	C	2.5	BC	9.5	BC	11
	Agent 2:(0 10 2 17 2 15 9 21)	AB	3	A	1.5	A	1
	Agent 3:(0 9 3 14 5 16 10 20)		0		0		0
15	Agent 1:(0 4 3 8 2 11 10 17)		0		0		0
	Agent 2:(0 4 3 8 7 15 12 21)		0		0		0
	Agent 3:(0 4 7 13 8 13 17 24)	ABC	8.5	ABC	3	ABC	7
16	Agent 1:(0 1 7 12 3 6 14 18)		0		0		0
	Agent 2:(0 8 6 19 8 17 19 30)		0		0		0
	Agent 3:(0 9 7 19 10 24 19 34)	ABC	8.5	ABC	4.5	ABC	4
17	Agent 1:(0 9 9 19 2 13 12 25)	ABC	2.5	AB	2	AB	5
	Agent 2:(0 5 3 11 7 17 15 22)		0	C	1	C	4
	Agent 3:(0 3 8 14 3 11 16 21)		0		0		0
18	Agent 1:(0 9 10 20 9 20 23 33)	ABC	18	ABC	3.5	ABC	7
	Agent 2:(0 9 6 16 2 12 11 22)		0		0		0
	Agent 3:(0 3 10 17 7 15 21 26)		0		0		0
19	Agent 1:(0 1 6 10 6 9 15 19)	C	2		0		0
	Agent 2:(0 4 7 14 10 18 22 29)		0	ABC	9.5	ABC	10
	Agent 3:(0 5 4 14 1 10 6 16)	AB	2.5		0		0
20	Agent 1:(0 10 8 22 7 18 18 31)		0	ABC	5	ABC	5
	Agent 2:(0 6 10 17 4 15 18 26)		0		0		0
	Agent 3:(0 9 8 18 3 16 12 26)	ABC	1.5		0		0

Figure A.1: Sample data for 20 Problems