

An Approximate Algorithm for Resource Allocation using Combinatorial Auctions

Viswanath Avasarala, Himanshu Polavarapu, Tracy Mullen
School of Information Science and Technology
Pennsylvania State University
{vavasarala, tmullen}@ist.psu.edu
himanshu@psu.edu

Abstract

Combinatorial Auctions (CAs), where users bid on combination of items, have emerged as a useful tool for resource allocation in distributed systems. However, two main difficulties exist to the adoption of CAs in time-constrained environments. The first difficulty involves the computational complexity of winner determination. The second difficulty entails the computational complexity of eliciting utility valuations for all possible combinations of resources to different tasks. To address both issues, we developed a new algorithm, Seeded Genetic Algorithm (SGA) for finding high quality solutions quickly. SGA uses a novel representational schema that produces only feasible solutions. We compare the winner determination performance of our algorithm with Casanova, another local stochastic search procedure, on typically hard-to-solve bid distributions. We show that SGA converges to a better solution than Casanova for large problem sizes. However, for many bid distributions, exact winner determination using integer programming approaches is very fast, even for large problem sizes. In these cases, SGA can still provide significant time savings by eliminating the requirement for formulating all possible bids.

1. Introduction

An auction can be defined as “a market institution with an explicit set of rules determining resource allocation and prices on the basis of bids from the market participants” [1]. Market-oriented programming uses market mechanisms like auctions to build computational economies to solve distributed resource allocation problems. In standard auction mechanisms, individual goods are auctioned independently of each other. These mechanisms can lead to inefficient allocations when agent utilities for various goods show strong complementarities. Combinatorial auctions allow agents to bid on a combination of items, and express any complementarities directly. For example, consider resource allocation in a sensor network, where the available resources must be allocated to different target tracks. To maintain a target track, not only should measurements be scheduled over the various sensors, but bandwidth for measurement

communication to the fusion system should also be reserved. Scheduling measurements will not be of any use to the consumers unless bandwidth to communicate the measurements is also available. Combinatorial auctions can be used in such environments to allow bidders to express such synergistic relationships between goods. Combinatorial auctions have been used for resource allocation in a number of domains with strict real time constraints, like sensor management (SM) [2, 3], supply chain management [4] and computer grids [5]. However, two issues have acted as stumbling blocks to the adaptation of CA in time constrained environments. The first is the computational complexity of finding the optimal allocation in the combinatorial auction, aka winner determination problem, which is NP-hard [6]. The second is the computational complexity of eliciting valuations for all possible combination of resources to different tasks/users. In this paper, we propose approximate algorithms that reduce the overall computational complexity of using CA’s for resource allocation to polynomial run times.

Recent years have seen great strides in the research of winner determination in combinatorial auctions. Depending on the distribution from which the bids are generated, either the algorithm CABOB [7] or Integer Programming approaches using a standard commercial software package such as CPLEX [8], have been found to give the best real-time performance. For many realistic distributions, CPLEX and CABOB perform extremely well with average running times of less than a second for problem sizes with thousands of bids. However, for certain complex bid distributions, neither CPLEX nor CABOB’s real-time performance may be adequate in environments with strict real-time constraints. For example, we tested CPLEX and our implementation of CABOB on these “hard” bid distributions” (see [7] for description of various bid distributions) with a problem size of 1000 bids and recorded up to 10,000 seconds running time. Note that our version of CABOB may not be as efficient as the original, but we are mainly looking at order-of-magnitude effects. Casanova, a local stochastic search procedure, is an approximate winner determination algorithm, based on Novelty+ [9, 10]. Casanova gives good quality solutions, very quickly even for fairly large problem instances on a variety of bid

distributions. However, in our experiments, we found that Casanova’s performance falls significantly with increase in problem sizes.

Standard winner determination algorithms require bids on sets of resources as input. But bid-formulation is also potentially computationally expensive [11, 12]. For example, if there are n tasks and m goods, $n \cdot (2^m - 1)$ bids may have to be formulated for computing the optimal allocation. Researchers have experimented with many approaches to tackle the complexity of bid formulation aka preference elicitation (see [13] for an exhaustive survey). However, except under very restrictive assumptions, preference elicitation is exponential in complexity. Particularly, in technical domains like sensor fusion where human participation in bidding process is absent, the cognitive limitations that prevent buyers from bidding on exponentially-many bundles are also not present. In these domains, restricted preference elicitation techniques do not work.

To overcome the above mentioned shortcomings, we formulated a novel genetic algorithm based search procedure, SGA (Seeded Genetic algorithm), for approximate winner determination in combinatorial auctions. This paper is organized as follows. Section two presents the algorithm details. In the results section, we compare the performance of SGA with other winner determination algorithms, both in terms of real-time performance and optimality. The final section describes our conclusions and directions for future work.

2. SGA description

This section provides an explanation of the winner determination algorithm and provides a brief overview of some previous algorithms, before discussing SGA in detail.

2.1. Winner determination for resource allocation using CAs

Assume that the auctioneer has a set of m goods $G = \{g_1, \dots, g_m\}$ to sell, and a set of n bids $B = \{b_1, \dots, b_n\}$. Bidders offer bids of the form $B_i = \langle b_i, p_i \rangle$, where b_i is the bundle of goods and p_i is the price the bidder is willing to pay for. The winner determination problem is to find an allocation of goods that maximizes the overall utility, given the constraint that each good can be sold to no more than one bid. Formally, this problem can be characterized as [6]:

$$\max \sum_{j=1}^n p_j x_j \quad s.t. \quad \sum_{j|i \in S_j} x_j \leq 1, \forall i \in \{1..m\}$$

where x_j is 1 if the bid is accepted to the final allocation and 0 otherwise.

This problem can be solved as a mixed integer programming problem using standard MIP software like CPLEX. Another example of exact winner determination procedure is CABOB, which uses a specialized depth first search that branches on bids. CABOB constructs a *bid-graph* with each bid in the set of bids B as the vertex of the graph and edges between vertices only when they have items in common. CABOB then uses a separate depth-first search (DFS) to find the optimal solution. By selecting an initial bid as being IN a possible allocation, then remaining bids with items that do not overlap this bid can also be IN the allocation and are considered for allocation in a particular search path of the DFS. CABOB bounds the DFS by pruning paths that have already been dominated.

The approximate winner determination search procedure, Casanova, is based on local stochastic search. Casanova starts with an empty allocation and bids are chosen from the unsatisfied bids randomly and added to the allocation vector. The probability that a bid is chosen is determined using its normalized score, calculated as the bid price divided by the product of the number of goods the bid consumes, the bid’s age, and the number of steps since it has last been chosen to be added to the allocation vector.

2.2. SGA

Genetic algorithms (GAs) are stochastic and polynomial in the number of bids, rather than exponential. To design a fast genetic algorithm based algorithm for combinatorial auctions, we initially experimented with a binary string representation for the GA chromosome. In this representation, a solution is coded as a binary string with length equal to the number of bids in the problem instance. A binary string is decoded into an allocation vector as follows: a bid is labeled *winning*, if the corresponding string bit is one and as *losing*, otherwise. However, this schema produces infeasible solutions where multiple bids containing a common item are marked as winning. The genetic algorithm has poor convergence, in spite of the use of penalty functions to penalize infeasible allocations. To overcome this problem, we devised a novel representational schema based on a ranking scheme that produces only feasible solutions that is described below.

2.2.1. Representational Schema. In the ranking-based representation schema, a chromosome is represented by a string of numbers, where each number corresponds to the rank of the corresponding bid. The solution represented by a chromosome is decoded in the following manner:

1. Initialize solution with the lowest ranking bid.
2. Select bid with the next highest rank. Add it to the solution if it does not create an infeasible solution, otherwise, proceed to the next highest ranking bid.

3. Continue till no bid can be added without creating an infeasible solution.

For example, consider the following scenario with 4 bids and 4 items (table 1). Decoding the allocation vector associated with the chromosome “2-1-3-4” is done as follows. The initial allocation vector is empty. Since, the lowest ranking bid is bid 2 (with rank 1), bid 2 is added to the allocation vector. The next lowest ranking bid is bid 1 (rank 2). However, bid 1 has item *a*, which is also present in bid 2. Since bid 2 has already been labeled as winning, bid 1 is labeled as losing. The procedure is repeated until the status of the highest ranking bid (bid 4 in this case) is determined. Thus, the above chromosome corresponds to a solution where bids 2 and 4 are marked as winning and the rest are marked as “losing”.

Chromosome : 2-1-3-4		
Bid No.	Items	State
1	{a}	Lose
2	{a,b}	Win
3	{b,c}	Lose
4	{c,d}	Win

Table 1. Example Scenario

2.2.2. SGA operators. We used the standard *crossover*, *mutation* and *tournament selection* operators in the genetic algorithm. The crossover operator produces two offspring chromosomes by swapping randomly chosen crossover segments between two parent chromosomes. The mutation operator changes the rank of a randomly chosen bid to a random value. The tournament selection operator replaces all the chromosomes in a given group with equal number of replicas of the chromosome with the highest fitness from the group.

The starting population used in SGA, consists of strings in which the integers, 1 to the number of bids, are randomly allocated to the different bids. In addition to these operators, to prevent degeneracy in the chromosome population, we added an operator called *regenerator*. To understand the requirement for this operator, consider the crossover operation between the following two chromosomes of length four:

- a. chromosome A: 2--1--3---4
- b. chromosome B: 3--2-- 4---1

If the crossover segment is of length two and the crossover point is chosen to be two, the offspring chromosomes produced from the crossover operation are

- c. chromosome C1: 2--2--4---4
- d. chromosome C2: 3--1--3---1

As shown, mutation and crossover operators lead to having multiple copies of the same bid in a chromosome. These chromosomes can be decoded using heuristics. For

example, assign a unique number *bidID*, to each bid. If two bids have the same rank, only add the bid with the lower *bidID* to the allocation vector. However, as the genetic algorithm evolves and the number of generation increases, the degeneracy in the chromosomes increases. The genetic algorithm might then prematurely converge, since the chromosomes lose the strength to produce high quality solutions with limited variability in the bid ranks. To avoid this problem, we devised the *regenerator* operator. The regenerator takes a *degenerate* chromosome and produces a *healthy* chromosome, in which no rank is repeated. The algorithm used by the regenerator operator is as follows:

```

i = 0; r = 1;
do while (r < no. of bids) {
  i = i + 1;
  B = set of bids with rank == i;
  while ( B is not empty) {
    b = bid with lowest bidID from B
    rank of b = r;
    remove b from B;
    r = r + 1;
  }
}

```

The computational complexity of the regenerator operator is linear in the length of the chromosome. The regenerator operator ensures that each bid in the chromosomes has a unique rank and these ranks are distributed between one and the number of bids in the problem instance. This has an added advantage. Regeneration makes the computational complexity of chromosome sorting for the purpose of decoding the allocation associated with it linear in the length of the chromosome. This greatly increases the speed of the genetic algorithm.

2.3. Seeding the GA

If an initial high quality solution is available, the solution can be used to seed the genetic algorithm by initializing some of the initial random population to the high quality solution. A seed chromosome, *seed*, is created from an initial solution as follows

```

n = size of initial solution.
for each bid b
  if(b belongs to the initial solution)
    seed[b] = random number between n
    and (no of bids + 1)
  else
    seek[b] = random number between 1
    and (n + 1)
Regenerate(seed)

```

If the GA is seeded with a high quality initial solution, it converges very rapidly, often improving on the initial seed. Seeding has two utilities:

1. If a faster, but less efficient approximate winner determination algorithm is available, the solution provided by this algorithm can be used as a seed for the GA, enabling it to converge more rapidly.
2. Seeding can be used for approximate incremental winner determination. Assume that the auctioneer has calculated an optimal (or a high quality solution) for the bids it has received. If certain bids are retracted or new bids are added (as can happen in the SM scenario), the auctioneer can use the original solution as a seed to obtain high quality solutions to the new problem very quickly.

Linear programming can be used to obtain a quick lower bound for the winner determination problem [7]. For certain bid distributions, this lower bound gives a very quality solution. For example, for CAT distributions and weighted random distribution [7], the linear programming solution is usually close in value to the optimal allocation.

2.4 Avoiding Explicit Bid Formulation

Consider a resource allocation problem with m resources $R = \{r_1 \dots r_m\}$ and n tasks $\{t_1 \dots t_n\}$. Let $\Theta = \{\theta_1, \theta_2, \dots\}$, be the power-set of R . To use standard combinatorial auction based approach for allocating resources to the various tasks, bids of the form $b_{ij} < \theta_i \rightarrow t_j, u_{ij} >$ where u_{ij} is the utility of allocating resource set θ_i to task t_j should be formulated as a preprocessing step. This is infeasible when the number of items to be allocated is large and the real-time constraints are strict. The representation schema used by SGA can be modified to avoid explicit bid formulations. The representation schema used by SGA, for these problems, is as follows:

The chromosome is represented by a string of length m . Each chromosome member is a number between 1 and n . For example, consider a three-task, four-item example. The chromosome "2-1-3-2" represents an allocation in which the bid on task II with items one and four, bid on task I with item 2 and bid on task III with item 4 are labeled as 'winning'. For evaluating the fitness of the chromosome, only the utilities of these three bids need to be calculated.

If the size of the population used in the SGA is λ_1 and the number of generations that SGA is run for is λ_2 , the total number of bids that need to be evaluated for this representation schema is $\lambda_1 * \lambda_2 * m$. For the standard formulation, $n(2^m - 1)$ utility computations are required. In a distributed environment, where task utility information is not available to the resource allocating algorithm, SGA involves the users submitting $\lambda_1 * \lambda_2 * m$ bids to the

auctioneer. To prevent this communication load, users could be made to submit their utility function, $\varphi_i(S)$ as their bid at the start of the auction, where $\varphi_i(S)$ is the i -th user's reported utility for resource set S . This bidding procedure is similar to General Vickrey Auction mechanism [14]. Approximate techniques like function-estimation neural-networks can be used to evaluate $\varphi_i(S)$ in polynomial run-time, when exact formulation is computationally expensive. For illustration of how machine learning algorithms have been used to estimate $\varphi_i(S)$ in sensor networks, using domain-specific knowledge, refer to [15]. It should be noted that there is no straightforward method to adapt Casanova to avoid an exhaustive bid formulation stage.

3. Results

We performed two sets of experiments to validate the performance of SGA. The first set of experiments compares the performance of SGA and Casanova on *hard* bid distributions, which are difficult for exact winner determination algorithms (see section 3.1 for details). The second set of experiments compares SGA's performance to CPLEX on *soft* bid distributions, which are easy for exact winner determination techniques (see section 3.2 for details).

3.1 Comparison with Casanova

For our experiments on hard distributions, we selected two distributions from Sandholm's bid distributions [8] that proved to be the toughest for the exact winner determination algorithms, which are

- a. Uniform (n, m, λ) distribution which consists of n bids, each with λ items chosen without replacement from the m items. Price is chosen randomly from a uniform distribution on $[0, I]$. For our experiments, we used $m = n/10$ and $\lambda = 5$, as in [7].
- b. Bounded $(n, m, \lambda^1, \lambda^2)$ where the problem consists of n bids. The number of items λ is randomly chosen between a lower bound λ^1 and an upper bound λ^2 . The price is chosen from a uniform distribution on $[0, \lambda]$. For our experiments, we used $m = n/10$ and $\lambda^1 = 1$ and $\lambda^2 = 5$, as in [7].

These distributions were the hardest for both CABOB and CPLEX. For example, CPLEX took an average of 4300 (757) seconds on a 2.8 GHz Pentium IV processor for problem sizes of 900 bids for uniform (bounded) distributions. For larger problems, the run time was much higher. For a sample run, a problem size of 1000 bids

with uniform distribution took more than 13 CPU hours. The parameters used for SGA are shown in table 2.

Population size (> 1400 bids)	6500
Population size (<= 1400 bids)	6000
Tour size	2
Crossover probability	0.995
Mutation probability	0.02

Table 2. SGA parameters

Our implementation of CABOB was slower than CPLEX for the above distributions. This is consistent with the reported results for CABOB and CPLEX [7]. We used the same parameters for all the experiments, except the population size. For problem sizes with less than 1400 bids, we used a population size of 6000 chromosomes and for larger problems, we used a population size of 6500. For Casanova, we used the parameters suggested by the authors in the original paper. However, we found that changing the walk probability to 0.4 gives better results and hence $w_p = 0.4$ was used. We tested both Casanova and SGA on large problem sizes, varying the number of bids from 1000 to 2000 bids.

To accurately compare SGA's real-time performance to the optimal solution requires first finding the solution using an exact winner determination algorithm. However, given our large problem sizes, the exact winner determination algorithms took days to find the optimal solution. Instead, we adopt a statistical approach to estimate the quality of the solutions generated by SGA and Casanova. We postulated that the average optimality for a particular distribution and a given problem size (fixed number of bids and items) is directly proportional to the number of items in the auction. We tested this hypothesis for problem sizes of 500 to 800 bids for both distributions using the following approach. For each problem size, we generated 20 random problems and calculated the average value of the optimal solution. The R-square of the least squares regression line with the problem size (number of bids) as the estimator and the average optimality as the regressor was greater than 0.99 for both the distributions. Hence, the linear relation between the estimator and the regressor is fairly strong. Figure 1 (Figure 2) shows the regression line and the optimality values of all the points generated for the uniform (bounded) distribution. We did not notice any particular trend in the variation of the optimality from the mean optimality based on problem size. For calculating the average optimality of a given problem size (ϕ_{size}), we extrapolate the regression lines in Figure 1 (Figure 2).

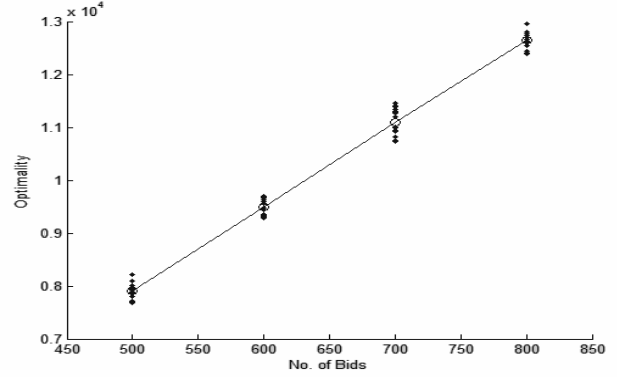


Figure 1. Regression line for average optimality versus problem size for uniform distribution.

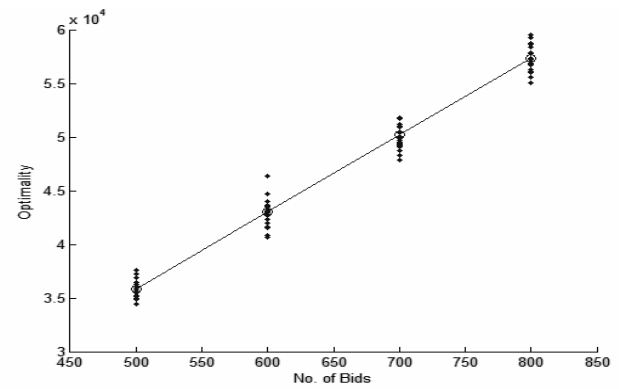


Figure 2. Regression line for average optimality versus problem size for bounded distribution.

This extrapolation is reasonable, since there was no increasing trend in variation of the optimality from the mean optimality with increase in problem sizes. Figures 3 (Figure 4) gives the average estimated optimality, obtained using SGA and Casanova on uniform (bounded) bid distribution. To generate these results, SGA and Casanova were run on 50 different randomly selected problems, for each problem size. The average optimality of the problems (ϕ_{size}), is calculated by extrapolation using Figure 1 (Figure 2). For calculating the percentage optimality yielded by SGA and Casanova, we divide the mean outcome of the approximate algorithms for a particular problem size with ϕ_{size} . Since these results are not exact and depend on the robustness of the extrapolation method used, we also show the 95% confidence interval of the optimal solution for each problem size. The 95% confidence interval is calculated using the 95% confidence interval for ϕ_{size} . This is calculated assuming that standard deviation of ϕ_{size} does not depend on problem size (which is consistent with our experiment results). Figures 1 and 2 show that Casanova is better for smaller problems, while SGA gives better results for larger problem sizes. This is because Casanova

search is primarily memory-less. It discards solutions generated in the past periodically and restarts search every few seconds. Though the highly greedy approach used by Casanova is advantageous for smaller problems, a memoryless search does not work as well for larger problem sizes. Since SGA and Casanova are stochastic algorithms, we also show the scatter plot of the results obtained for a particular size for bounded distribution in Figure 5. For a large percentage of problems, we find that SGA gives better results than Casanova.

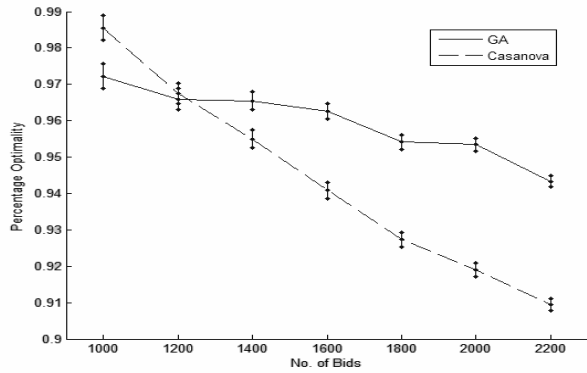


Figure 3. Estimated percentage optimality (with their 95% confidence intervals) versus problem size for uniform distribution. We used a cut-off time of 200 CPU-Sec on 2.8 GHz Pentium IV processor.

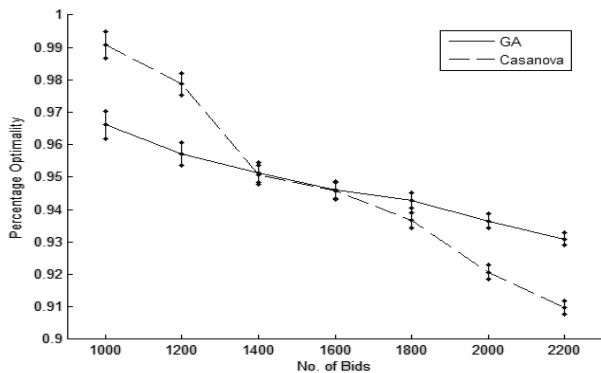


Figure 4. Estimated percentage optimality (with their 95% confidence intervals) versus problem size for bounded distribution. We used a cut-off time of 200 CPU-Sec on 2.8 GHz Pentium IV processor

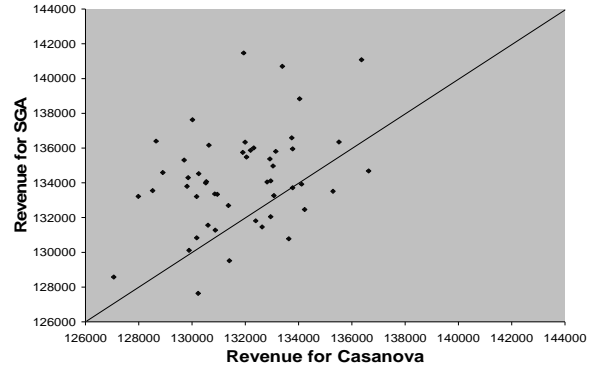


Figure 5. Correlation of revenue obtained by SGA and Casanova for bounded distribution with problem size of 2000 bids. The line shows of ratio of 1 when comparing the revenues.

3.1.1. Real-Time Performance. We compared the real-time performance of SGA and Casanova for the uniform bid distribution with problem size = 2000 bids (Figure 6). We found similar results for the bounded distribution. The highly greedy approach used by Casanova, results in better real-time performance initially. SGA starts with a randomly initialized population and evolution of good quality solutions requires a fair amount of computation. However, SGA is better than Casanova given longer time scales and converges to better optimality. The advantages of SGA and Casanova can be combined by using Casanova to seed SGA. Casanova can be made to run for sufficient time, till it reaches a plateau in its performance. The high quality solutions provide by Casanova can then be used to seed SGA. Figure 6 shows the real-time performance of SGA and Casanova when run separately. Figure 7 shows the comparative real time performance of Casanova and SGA, when Casanova is used to seed SGA (at $t = 70$ CPU-Secs).

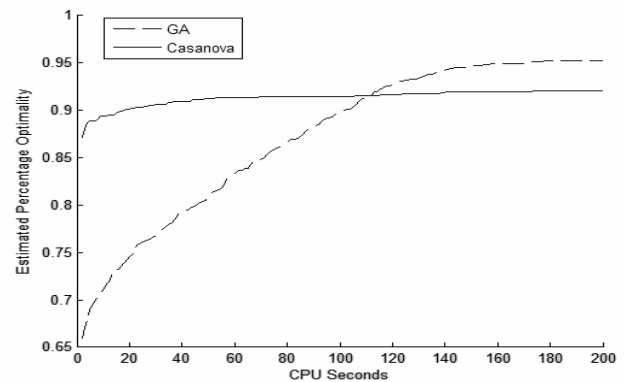


Figure 6. Real-time performance of GA and Casanova on a 2.8GHz Pentium-IV processor (averaged over 20 runs)

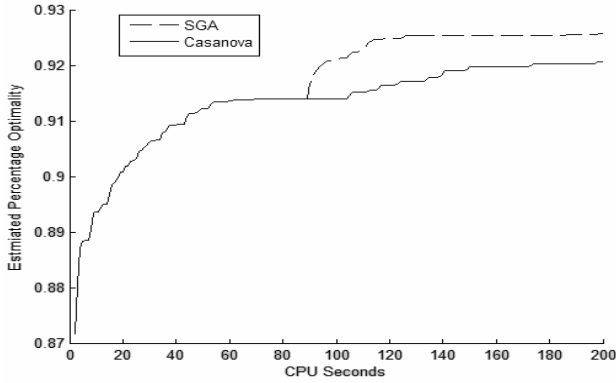


Figure 7. Real-time performance of SGA (seeded with Casanova) and Casanova on a 2.8GHz Pentium-IV processor (averaged over 20 runs)

As Figure 7 shows, the combined algorithm has better, or at least equivalent, real-time performance for all times when compared to Casanova and also converges to better optimality. It is interesting to note that SGA as a stand-alone algorithm has better overall convergence than SGA seeded with Casanova. This happens because the high quality solutions provided by Casanova lead to premature convergence in SGA.

3.2 Comparison to CPLEX

For many realistic bid distributions, exact winner determination using CPLEX based IP formulation is very fast, even for large problems [7, 8] to test performance of SGA on such “soft” bid distributions, we used the resource allocation problems generated by MASM [2, 3]. MASM is a market-based sensor resource allocation mechanism where the sensor manager generates combinatorial bids for every possible combination of sensors for each task. Due to space constraints, details of MASM and the underlying resource allocation problem are not explained, but are available in [15]. The time-taken for bid-formulation and for winner determination using CPLEX are given in Figures 8 and 9 respectively. Clearly, the bottleneck of using CAs in this domain is the bid formulation step that maps from tasks to sensor resources.

For SGA, a population size of 150 and the number of generations equal to (#bids/1000) was used. The other parameters are as shown in table 2. Figure 10 shows the comparison of the total time required for resource allocation using the CPLEX-based approach and SGA-based approach. Figure 11 shows the average optimality SGA achieved using CPLEX. Clearly, for a modest loss of optimality, SGA provides significant savings in real-time requirements of the allocation algorithms.

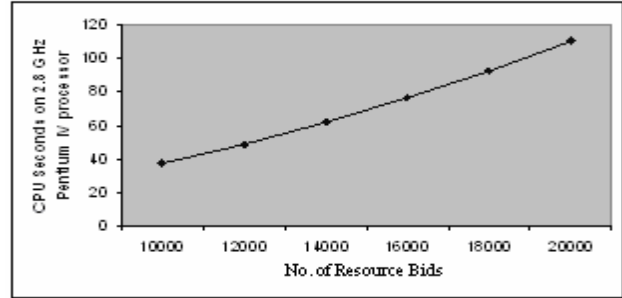


Figure 8. Time required for bid formulation (averaged over 10 runs)

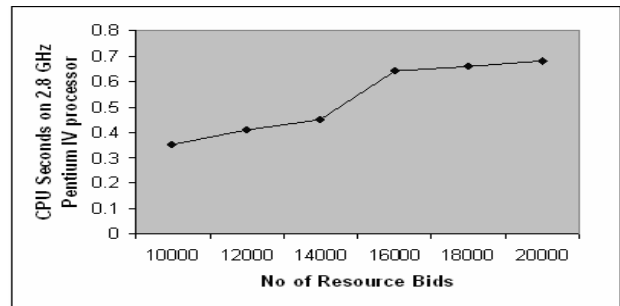


Figure 9. Time required for winner determination using CPLEX (averaged over 10 runs)

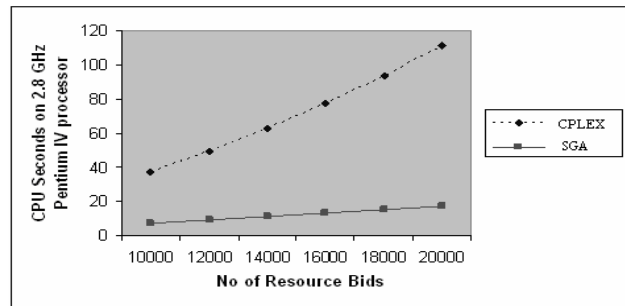


Figure 10. Comparison of time requirements of SGA and CPLEX-based optimization (averaged over 10 runs)

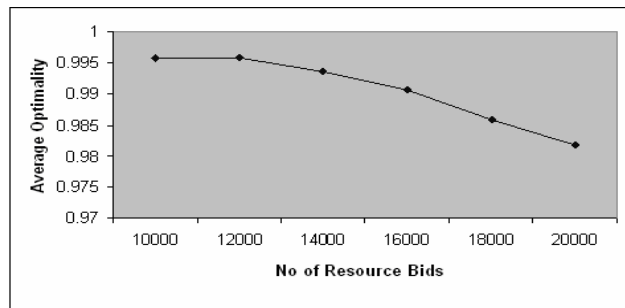


Figure 11. Optimality achieved by SGA (averaged over 10 runs)

4. Conclusions

The successful application of combinatorial auctions to environments with very strict real-time constraints is contingent on the development of adequate winner determination algorithms. Recent years have seen great strides in this area. Optimal winner determination for fairly large problem instances is now possible within a fraction of a second for large problems over various bid distributions. However, if the problem instances are too large or if the bid distributions are too difficult for the complete algorithms, approximate real-time algorithms are alternatives.

We implemented SGA for real-time approximate winner determination in combinatorial auctions and compared its performance with a local stochastic search procedure, Casanova. For the tested problem instances, we found that SGA, on average, converges to better solutions for the larger problem instances. Thus, SGA is a better candidate solution for pruning search in complete winner determination algorithms. Regarding real-time performance, we found that Casanova performs better in the initial stages but is later outperformed by SGA. Thus, the choice between the two algorithms depends on the real-time demands of the application domain. However, Casanova can be used to seed SGA to generate a hybrid algorithm that is better or at least similar to Casanova, for all time ranges.

The main advantage of SGA is that it does not require the set of all possible bids as an input. As a result, SGA provides significant time-savings over exact winner determination algorithms by avoiding the cost of bid formulation, as illustrated by the results from MASM.

A key issue with SGA is its use in strategic environments. Auction methodologies like GVAs are not necessarily incentive compatible when approximate winner determination techniques are used [16]. Also, in strategic environments, privacy concerns might prevent agents from revealing their valuation information [14]. This could be a problem if SGA requires users to submit their utility functions as bids. We are currently investigating techniques to add incentive compatible measures to approximate winner determination algorithms.

10. References

- [1] P. R. Wurman, M. P. Wellman, W. E. Walsh, "A parameterization of Auction Design Space", *Games and Economic Behavior*, Vol. 35, 2001, pp 304-338.
- [2] V.Avasarala, T.Mullen., D.L.Hall., and A.Garga, "MASM: market architecture or sensor management in distributed sensor networks", *SPIE Defense and Security Symposium*, Orlando FL, ,2005, pp 5813 – 30.
- [3] T.Mullen, V.Avasarala., D.L. Hall, "Customer-Driven Sensor Management", *Intelligent Systems, IEEE [see also IEEE Intelligent Systems and Their Applications]* Volume 21, Issue 2, March-April 2006, pp 41 – 49.
- [4] W.E.Walsh, M. Wellman and F.Ygge, "Combinatorial Auctions for supply chain formation", *In Proc. of ACM Conf on Electronic Commerce (EC'00)*, Oct 2000, pp 260-269.
- [5] A.Das, D.Grosu, "A Combinatorial Auction-Based Protocols for Resource Allocation in Grids", *19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05)*, 2005.
- [6] M.Rothkopf, A.Pekec, and R.Harstad, "Computationally manageable combinatorial auctions", *Management Science*, Vol. 44 No.8, 1998, pp 1131-1147.
- [7] T.Sandholm., S.Subhash., A.Gilpin, and D. Levine, "CABOB: A fast optimal algorithm for winner determination in combinatorial auctions", *proc. IJCAI*, 2001, pp. 1002-1008.
- [8] A.Andersson, T.Mattias, Y.Fredrik, "Integer programming for combinatorial auction winner determination", *In Proc. Fourth International Conf. Multi-Agent Systems (ICMAS)*. IEEE Computer Society, Boston, MA, 2000, pp 39-46.
- [9] H. Hoos, "On the run-time behavior of stochastic local search algorithms for SAT", *In Proc. AAAI-99*, 1999, pp 661-666.
- [10] H. Hoos and C. Boutilier, "Solving combinatorial auctions using stochastic local search", *In Proc. AAAI-00*, 2000, pp 22-29.
- [11] D. Parkes, "Optimal auction design for agents with hard valuation problems", *In Agent-Mediated Electronic Commerce workshop*, Stockholm, Sweden, 1999.
- [12] K.Larson and T.Sandholm, "Costly valuation computation in auctions", *In Theoretical Aspects of Rationality and Knowledge VIII*, Siena, Italy, 2001, pp 169-182.
- [13] V. Conitzer and T. Sandholm and P. Santi, "Combinatorial Auctions with k-wise dependent Valuations", *In Proc. of the National Conference On Artificial Intelligence (AAAI)*, Pittsburgh, PA, 2005, pp 248-254.
- [14] M. Rothkopf, T. Teisberg, and E.Kahn. Why are Vickrey auctions rare? *Journal of Political Economy* Vol. 98, No. 1, 1990, pp.94-109.
- [15] V.Avasarala, "Multi-agent systems for data-rich, information-poor environments", *PhD Thesis*. College of IST, Pennsylvania State University, 2006.
- [16] Lehmann, D., L. O'Callaghan, and Y. Shoham, "Truth Revelation in rapid, approximately efficient combinatorial auctions", *In Proc. 1st ACM Conf. on Electronic Commerce (EC-99)*, 1999, pp. 96-102.