A Non-exact Approach and Experiment Studies on the Combinatorial Auction Problem

Y. Guo[1], A. Lim[2], B. Rodrigues[3] and Y. Zhu[2]

[1] *Department of Computer Science, National University of Singapore*
3 Science Drive 2, Singapore 117543
guoyunso@comp.nus.edu.sg

[2] *Department of IEEM, Hong Kong University of Science and Technology*
Clear Water Bay, Kowloon, Hong Kong
{iealim,zhuyi}@ust.hk

[3] *School of Business, Singapore Management University*
469 Bukit Timah Road, Singapore 259756
br@smu.edu.sg

Abstract

In this paper we formulate a combinatorial auction brokering problem as a set packing problem and apply a simulated annealing heuristic with hybrid local moves to solve the problem. We study the existing exact and non-exact approaches to the problem and analyze the performance of those approaches. We compared our heuristic with the leading exact method CPLEX 8.0 solver and another non-exact algorithms Casanova using both the CATS test sets and test cases believed more difficult than CATS. Results show that the method is competitive with CPLEX 8.0 and obtains near optimal solutions for the CATS cases and up to 15% and 40% better solutions compared with CPLEX and Casanova, respectively, when the other instances were used.

1. Introduction

Interest in combinatorial auctions has increased in the field of artificial intelligence, where a number of problems and their solutions have been studied. Once such auction problem can be described as follows: a single supplier has a number of jobs to fulfill and a number of bidders put forward a set of bids, each covering a subset of these jobs. Each job can be contained in at most one selected bid where each bid, if selected by the supplier, results in a profit to the supplier. The problem then is how the supplier should select bids which result in the best total profit. Recent work has modelled the auction problem as a Set Packing Problem (SPP), which is a well-known NP-hard problem [11], [2], [1], [10], [4], [7]. The problem can be abstracted as follows: Assume there are n bids and m jobs and each bid can cover a number of jobs resulting in a profit to the supplier, w_j ($j \in 1, ..., n$) if bid j is selected, and $[a_{ij}]_{m \times n}$ is a m-row, n -column 0-1 matrix, where $a_{ij} = 1$ if job i is included in bid j. Further, the decision variables, $x_j = 1$ if the supplier selects bid j, and 0 otherwise. The integer programming model for the problem as a SPP is then:

$$\text{maximize} \sum_{j \in N} w_j x_j \tag{1}$$

Subject to:

$$\sum_{j \in N} a_{ij} x_j \leq 1, \; i \in M \tag{2}$$

$$x_j \in \{0, 1\}, \; j \in N \tag{3}$$

where $N = \{1, \ldots, n\}$, $M = \{1, \ldots, m\}$.The first set of constraints ensure that each row is covered by at most one column and the second integrality constraints ensure that $x_j = 1$, if and only if column j of the matrix is in the solution, i.e. bid j is selected.

Exact algorithms, such as branch-and-bound[2] and iterative deepening A* search[10], have been developed and applied to real world instances. Optimal solutions have also obtained by the Ilog CPLEX solver [1]. In addition, non-exact algorithms, including an iterative greedy approach [7] and stochastic local search [4], have been studied in the literature. As combinatorial auction problems are of high relevance to applications, mechanisms for generating realistic test suites have been studied and a standard benchmark test set generator Combinatorial Auction Test Suite (CATS) has been developed [9].

In [10], Sandholm develops a bidtree IDA* search as an early exact algorithm for the SPP. Some good preprocessing techniques are also discussed such as partitioning and the removal of non-competitive bids. Sandholm's approach solves problems with several hundreds of bids and jobs in less than 10 hours. Another exact branch-and-bound algorithm called CASS on SPP for combinatorial auctions was proposed by Fujishima, Leyton-Brown and Shoham in [2]. The CASS algorithm is presented as a "naive brute-force approach followed by four improvements." [2]. The improvements include bids' bin concept, caching and a good bounding function. The algorithm exhibited fairly good performance providing optimal solutions for test sets with about one hundred jobs and thousands of bids in reasonable times. CASS outperforms Sandholm's approach with respect to the time [1]. The other exact algorithm makes use of CPLEX [1]. Although CPLEX is a broad tool, it offered good experimental results compared with Sandholm's IDA* search and Leyton-Brown *et al.*'s CASS. CPLEX can reach optimal solutions in shorter times for most test sets used in [1] generated by CATS. Recently, Sandholm proposed a second generation branch and bound based algorithm called CABOB [11]. In [1], CPLEX 6.5 shows better performance with respect of time spent than the IDA* algorithm and Sandholm benchmarks CABOB against CPLEX 7.0, which according to the author, is 1.6 times faster than CPLEX 6.5. Experiments in [11] show that CABOB is usually faster than CPLEX 7.0 for most CATS distributions used. According to ILOG specifications, CPLEX 8.0 is 40% faster than CPLEX 7.0. so that CPLEX 8.0 is a good benchmark for testing the SPP. We base our comparisons on CPLEX 8.0.

Besides exact algorithms, heuristic methods have also been applied to the SPP and extensive experimental studies have been carried out. In [7], a non-complete greedy heuristic was used which could find non-optimal solutions for several hundred of bids and jobs. In [4], a stochastic local search algorithm called Casanova reportedly outperformed CASS. It reports better result quality compared with CASS for test sets in [2] and [10] by setting cut off times for CASS and Casanova. Casanova bears a strong resemblance to the Novelty algorithm, one of the best-performing algorithms for solving hard SAT problems [4]. The stochastic search is based on scoring each search state using the "revenue per job" of the corresponding allocation [4]. The Casanova method is thus one of the leading non-exact algorithms we found in the literature.

In this paper, we introduce a new non-exact heuristic algorithm based on simulated annealing (SA) with hybrid neighborhood search techniques consisting of a branch-and-bound search, a greedy selection and a randomized 1,2-exchange. Preliminary results were obtained in [3] which improved on the iterative greedy search in [7]. We improved the algorithms proposed in [3] and studied the performance of our algorithm against the competitive exact and non-exact algorithms, namely the CPLEX 8.0 solver and Casanova, by applying them on test sets used in [1] which use four distributions from CATS. In addition, from [11] and our experimental results, we found the CATS test sets were solved by CPLEX and CABOB in a few seconds. Leyton-Brown *et al* discussed the empirical hardness of different CATS test set on the combinatorial auction problem in [8], which made it clear some CATS distributions are indeed trivial for CPLEX 7.1, while the uniform distribution is among the hardest distributions in CATS. Because of this, we have used another more difficult test set from [7] which factors in several real world constraints such as pricing, non-dominating bids (where bids that cover few jobs but provide large profit are called dominating bids and which do not usually occur in real situations). The results we obtain indicate that our new algorithm is good heuristic for the SPP.

This paper is organized as follows: in the next section, the simulated annealing algorithm is discussed and in section 3, experimental results are studied and compared. We conclude the work in section 4.

2. Simulated Annealing with Hybrid Local Moves Algorithm

Simulated Annealing (SA) is a meta-heuristic that differs from the traditional hill-climbing search in the sense that it may accept a down-hill move which can decrease the objective function value with a certain probability related to the temperature variable [6]. We used simulated annealing as our heuristic framework and applied various local moves to approach SPP (refer to Algorithm 1), with the local move probability constants p_1 and p_2 being 0.2 and 0.7 respectively, and the cooling factor C_0 is set to 0.9999.

Algorithm 1 Simulated Annealing Framework

$S \leftarrow \{\}$; $best\_value \leftarrow 0$;
$Temperature \leftarrow T_{max}$; $Iter \leftarrow 0$
$Preprocess\_on\_Bids()$
while $Iter < Max\_Iter$ and
$Temperature > T\_Terminate$ **do**
  with probability p_1
    $Stemp \leftarrow Branch\_and\_Bound\_Move(S)$;
  with probability p_2
    $Stemp \leftarrow Greedy\_Move(S)$;
  with probability $1 - p_1 - p_2$
    $Stemp \leftarrow 1, 2 - exchange(S)$;
  $\delta = value(Stemp) - value(S)$
  if $\delta \geq 0$ **then**
    $S \leftarrow Stemp$
  else {from S to $Stemp$ is a downhill move}
    $p = e^{-\delta/Temperature}$
    with probability p
      $S \leftarrow Stemp$
  end if
  if $value(S) > best\_value$ **then**
    $best\_value \leftarrow value(S)$;
  end if
  $iter \leftarrow iter + 1$
  $Temperature \leftarrow Temperature * C_0$
end while

2.1. Preprocesses on Bids

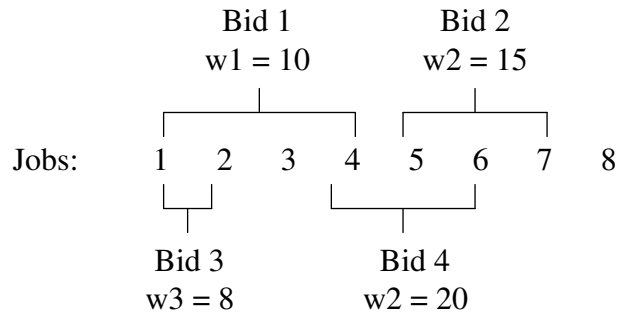
In order to increase the time efficiency of the algorithm, we should not consider bids that would definitely lead to suboptimal results. A simple example is, if two bids i and j, which cover exactly the same set of jobs, and $w_i \geq w_j$, we will never include bid j in the solution. Here we generalize this property by forming the problem RMV-n_1-n_2: Given a set of jobs J, and m bids $(b_1, \ldots, b_m)$, for all i, $b_i \subset J$. For the positive integers n_1,n_2, where $n_1, n_2 \leq m$. Can we find two sets of bids $S1$ and $S2$, such that the followings are satisfied:

- $\mid S1 \mid = n_1$ and $\mid S2 \mid = n_2$
- for all i, j, $i \neq j$, if $b_i \in S1$ and $b_j \in S1$ then $b_i \bigcap b_j = \emptyset$
- for all i, j, $i \neq j$, if $b_i \in S2$ and $b_j \in S2$ then $b_i \bigcap b_j = \emptyset$
- $\bigcup_{b_i \in S1} b_i \subset \bigcup_{b_j \in S2} b_j$
- $\sum_{b_i \in S1} w_i \geq \sum_{b_j \in S2} w_j$

If such two sets S1 and S2 can be found, we will not include the entire set of jobs S2 in the solution. Figure 1 is an example of RMV-2-2 where S1=$\{$bids3,bids4$\}$ and S2=$\{$bid1,bid2$\}$. The simple example we introduced at the beginning of this section is a specification of the RMV-n_1-n_2 problem, i.e. RMV-1-1.

If the problem RMV-n_1-n_2 has a solution, we can prevent the set of bids, namely S2 to be included in the solution and thus improve the efficiency of our heuristic. Unfortunately, the general RMV-n_1-n_2 is NP-Complete, since we can transform the Set Packing Problem to it.

One decision version of SPP can be expressed as: Given a ground set S, a collection of subsets C, each subset has a weight w_i, two constants W and K, can we find a collection of subsets $C' \subset C$ such that $|C'| = K$, $\forall c_1, c_2 \in C'$, $c_1 \cap c_2 = \emptyset$, and $\sum_{c_i \in C'} w_i \geq W$?



In this example bid_1 and bid_2 will never both be selected, as selecting bid_3 and bid_4 together would give a better result.

Figure 1. RMV-2-2 example

In the transformation, we add one bid $b' = J$(the entire set) and set the weight of b' to be W, and let $n_1 = K$, n_2=1. If there is a solution of RMV-K-1 with S2=b', we can conclude that the solution for SPP exists.

Since the problem is NP-Complete, no polynomial algorithms can be found unless P=NP. In the implementation, we only preprocess to handle RMV-1-1, RMV-1-2 and RMV-2-1, as general RMV-n_1-n_2 is computationally expensive.

In the preprocess stage to eliminate useless sets of bids, for the solutions obtained from RMV-1-1 or RMV-2-1, we directly remove the only bid in $S2$ from the candidate bid set. For the solutions obtained from RMV-1-2, we need to handle it differently. We will discuss this in the next section where the branch-and-bound search is introduced.

2.2. Local Move 1: Branch-and-Bound

The branch-and-bound search is a complete brute-force algorithm improved by adopting a suitable bounding function and other techniques to prune the search space, and giving results in an affordable amount of time spent. In [2], a branch-and-bound algorithm CASS is applied to the whole solution space. Therefore if given sufficient time, the exact algorithm can give optimal results for any test set. CASS is branching on the number of jobs, so it performs well on the test sets which have a very small number of jobs, usually 30 to 150, and a relatively large number of bids, which is usually several thousand. CPLEX IP Solver exhibits a better performance than CASS for most of the test sets generated from CATS, which are usually with several hundred jobs.

In our method, we employ the branch-and-bound idea to perform the local move by applying it to a small fragment of a solution. We select a subset of jobs and bids and try to find the optimal solution on this small portion, while keeping the other jobs and bids in the current solution unchanged. The details are presented in the followings.

2.2.1. Overview Let the current solution be the set of selected bids S, with $\mid S \mid = m_0$. Our branch and bound local search will randomly choose m_1 ($m_1 \leq m_0$) bids ($b_1, \ldots, b_{m_1}$) in S, and remove them from the solution. As each bid is a set of covered jobs, consider the set of jobs $\bigcup_{i=1}^{m_1} b_i$s as K. Let P be the maximum set of bids $c_1, \ldots, c_{|P|}$, such that $\forall c_i \in P$, $c_i \subset K$. Then, we use branch-and-bound to find the optimal solution for bids in P which is put back into S, the original solution set. Obviously, the new solution is feasible as all bids in P do not conflict with any other bid in S and is at least as good as the old one, since the partial optimal solution for bid set P is found by the branch-and-bound search. In the implementation, we chose a small m_1 so that the number of bids in P is small, and the branch-and-bound search can be fast and used as a local move in our SA framework.

2.2.2. Implementation Let P be defined as in the previous section. The aim is to find the optimal solution with respect to bids in P by branch-and-bound search. Several preprocessing techniques are used to improve time efficiency.

Suppose we represent P by a graph $G < V, E >$. Let V be the nodes representing each bid, $|V|=|P|$. Each edge in E from node i to node j represents the fact that bid i and bid j conflict with each other. We first partition G into connected components, where, clearly, the sum of optimal solutions of each connected component is the optimal solution for the bid set P. This preprocessing step is used in [10].

In order to handle RMV-1-2 discussed in the previous section, we simply add an edge between the two bids if they are in the same connected component, to make sure that the two bids will never be selected together; if the two bids are in different connected components, we will not add this edge because otherwise it would form a larger connected component and decrease efficiency.

The bounding function used is similar to the one used in CASS. For each job covered by any bid in P, we overestimate the profit each single job i can bring by: $t(i) = \max\{w_j/|b_j| : 1 \leq j \leq m, \text{job } i \in b_j\}$. Suppose the current solution S consists of m_0 bids ($b_1, \ldots, b_{m_0}$), then the bounding function is:

$$bound(S) = value(S) + \sum_i t(i)$$

where $value(S)$ is the total profit from S, and i is such that job $i \in b_j$ for some $b_j \notin S$ and job $i \notin b_k$ for all $b_k \in S$.

Whenever the bound is smaller than the current best solution, the current branch is discarded. We note that the branch-and-bound method used for a partial solution hybridize an exact search within a non-exact a local move heuristic.

2.3. Local Move 2: Greedy Local Search

The greedy local search will find an unselected bid that does not conflict with any bids already in the current solution and has the largest greedy value. In any greedy local search, the choice of the greedy value will have a great impact on the solution quality. The profit of a job is a good greedy value for a bid, but the profit w_i of bid b_i does not reflect the fact that the more bids b_i conflicts with, the more constraints will be incurred on the bids if include b_i is included in the solution. Hence, the greedy value of bid b_i is defined to be the profit from b_i less the penalty incurred by b_i, which we denote by $penalty_i$ defined by:

$$penalty_i = \sum_{j=1}^{n} (C_1 * profit_j - C_2 * \sum_{k=1}^{n} profit_k * c_{jk}) * c_{ij}$$

Here, $c_{ij} = 1$ if and only if there is a k such that $a_{ik} = 1$, where a_{jk} is defined in section 2, and C_1 and C_2 are scaling factors. $penalty_i$ for b_i takes into consideration the total profit of all bids that conflict with b_i, and also conflicting bids with b_i which conflicts with other bids.

2.4. Local Move 3: 1,2-Exchange

The 1-exchange and 2-exchange moves are random moves in the solution space. They randomly pick 1 or 2 unselected bids from the candidate bid set. For 2-exchange, it only selects 2 bids that are not in conflict with each other. It then removes any bid in the current solution that will conflict with newly selected bid(s), and adds the new 1 or 2 bids into the current solution.

3. Experimental Results

In this section, we present the experimental results and compare the method developed here (denoted

SAGII) with the exact CPLEX 8.0 solver and non-exact Casanova algorithm. CPLEX uses the SPP model provided in [1], and the parameters were tuned for the best solutions we could achieve. As we were unable to secure the original Casanova program from the authors of [4], we implemented Casanova as described in [4] and extensively tuned all the parameters. We tested the three approaches first using test sets generated by CATS which were used in [1] with four different distributions. Then, we tested the three approaches using test sets generated from Lau *et al.* [7] of various problem sizes, where generating mechanisms were different from CATS in the sense that [7] factors more real-world imputs such as bidders' pricing range, fairness among bidders and each job's preference with different bidders, etc. All the experiments were run on machines with Pentium IV-2.40GHz CPU, 1GB Memory.

3.1. Comparison Using CATS Test Set

In [9], Leyton-Brown *et al.* discussed the need to have a standard test suite for combinatorial auction problems, and analyzed the way for generating test suite with various emphasis through different mathematical distributions. CATS programs were developed for researchers to generate their own test sets with different problem sizes and distributions such as the binomial distribution and exponential distribution. So far, there have been a number of papers which used CATS as a standard test suite to measure performance, e.g., [5], [13], [12] . In our work, we used the test sets in [1]. We follow the name convention for test sets, e.g. UNI-p-q is a test set with uniform distribution, p jobs and q bids and applied four types of test sets, namely test sets with random, binomial, exponential and uniform distributions and analyzed the performance of CPLEX, Casanova and SAGII. The results are presented in Table 1, where t_1, t_2 and t_3 are the average time required for CPLEX, Casanova and SAGII, respectively, for the 10 instances of each category in seconds. μ measures the average result values of each method and $\delta_{Casanova}$ and δ_{SAGII} are the differences, in percentage, of the Casanova and SAGII results from optimal results obtained by CPLEX, respectively.

From Table 1, we see that SAGII always outperforms Casanova providing 20% to 30% improvement in results for all distributions in 1/6 to 1/4 of the time spent as Casanova. We believe this is partially because Casanova's local search is much more random and uninformed than SAGII. In [4], Casanova makes use of two kinds of local searches: a totally random move and a greedy local move, both of which do not closely relate to the properties of the SPP. In SAGII, however, in addition to random local moves, i.e. 1,2-exchange, guided local searches are used, namely branch and bound search and greedy local search. The greedy value used is more detailed than that used in Casanova. Besides that, our branch-and-bound local move hybridizes the exact local algorithm within the non-exact simulated annealing heuristic framework. This hybridization makes use of the property of SPP that when a partial optimal solution is found for a certain set of jobs, the new solution after replacing the corresponding part with the partial optimal solution, is also feasible.

Compared with CPLEX, for 3 out of the 4 distributions (random, binomial and exponential), SAGII gives results with good quality always within 5% from optimal. Note that SAGII produces optimal results for all test sets with binomial distribution, and the time spent is less than 1/6 the time required for CPLEX. However, SAGII did not produce results good enough compared with CPLEX for the uniform distribution, where CPLEX enjoys approximately a 15% advantage. According to [11], the uniform distribution is a hard test case for CABOB as well as requiring running times about twice the time spent of CPLEX 7.0. Casanova also obtained very poor results for uniform distribution instances — the results were more than 40% away from optimal.

In general, in the experiments, CPLEX was the best exact algorithm for CATS test set. But after analyzing log reports from CPLEX, we discovered that the four CATS generated test sets were generally "easy" for CPLEX as the number of Gomory Fractional Cuts CPLEX made was usually very small. This is also the reason why CPLEX ran extremely fast for the random and exponential distributions. Also from the generating mechanisms of CATS, which are mainly concerned with mathematical distributions on bids and prices, there is still room for the test suite to be improved by considering other realistic factors. We also compared the three methods on test sets from [7] in the next section.

3.2. Comparison Using Another Realistic Test Set

CPLEX, Casanova and SAGII were used to test sets generated from [7]. As in section 3.1, even more test sets with comparable number of jobs and bids are used. The test sets include up to 1500 jobs and 1500 bids. We find that these test sets are also realistic in the sense that they considered real auction factors such as bidder fairness, job's preference etc. It considers several factors in deciding the structure of a test set, such as a pricing factor which models a bidder's acceptable price range for each bid, a preference factor which models the bidder's preference in different bids, and a fairness factor which

Test set	# instance	t_1	μ^1_{CPLEX}	t_2	$\mu_{Casanova}$	δ_1	t_3	μ_{SAGII}	δ_2
RND-400-2000	10	4.2	16143.8	244.2	10505.4	34.93%	58.9	15950.0	1.19%
BIN-150-1500	10	198.6	109293.0	148.5	89546.8	18.07%	26.5	109293.0	0
EXP-30-3000	10	0.1	44723	446.3	32662.2	26.97%	81.0	43241.7	3.31%
UNI-100-500	10	25.9	129050.0	109.6	69153.3	46.41%	18.8	110941.5	14.03%

Table 1. Experimental results on CATS test set

measures the fairness in distributing of jobs among bidders.

We found that CPLEX performed more badly than when applied to CATS distributions. We compared the performance of CPLEX, Casanova and SAGII on some of the test sets from [7]. Test sets are labeled REL-n-m, where n is the number of jobs and m is the number of bids. Because CPLEX cannot give the optimal values for any of the test sets in a reasonable amount of time, in order to give advantage to CPLEX, we allowed CPLEX 3600 seconds to run each of the 15 test cases generated randomly from [7] and compared the results CPLEX obtains at that time, while both SAGII and Casanova terminate in much shorter times (see Table 2), where t_1 and t_2 is the time spent for SAGII and Casanova for each test instance, respectively and δ_1 and δ_2 measures the percentage of of SAGII over CPLEX and Casanova, respectively.

From the table, we see that SAGII consistently outperformed CPLEX giving better results in much shorter times for every instance. SAGII produced, on average, a 4.15% better results than CPLEX for the REL-1000-1000 test set, 7.53% for REL-1000-1500 test set and 9.24% for REL-1500-1500 test set. Note that SAGII outperforms CPLEX increasingly as test set size increased. We note that SAGII only used about 1/80 to 1/40 the time when compared with CPLEX in experiments.

Comparing SAGII with the non-exact algorithm Casanova, we can see that SAGII outperforms Casanova by providing a more than 30% better result in about half the time required by Cassanova. In the next section, we compare the two heuristic methods more intensively using 500 test instances generated by the mechanism described in [7].

3.3. Intensive Experimentation on SAGII and Casanova

500 instances are generated with 5 different problem sizes in order to compare the performance of SAGII and Casanova. Table 3 shows the results, where t_1 and t_2 are the average times for SAGII and Casanova, respectively and μ_{SAGII} and $\mu_{Casanova}$ are the average results of the two methods. δ measures the difference $(\mu_{SAGII} - \mu_{Casanova})/\mu_{SAGII}$.

When applied on the new test sets, Casanova failed to obtain the same-quality result as SAGII. In addition, we find that SAGII's performance is stable with respect to each 100 test instances in one category. It can always outperform Casanova as it usually obtains results which are more than 30% better than Casanova's results, and SAGII requires only about half the running time compared with Casanova. We note that SAGII also outperforms Casanova in the CATS test sets.

4. Conclusion

In this paper, we modeled a combinatorial auction problem as a set packing problem. We surveyed the literature for exact and non-exact algorithms developed for the problem and proposed a new heuristic method based a simulated annealing framework with three types of local move strategies, including a hybridization of exact branch-and-bound search in the non-exact simulated annealing heuristic. The results in extensive experiments showed that the method outperformed the leading heuristic, Casanova. When compared with CPLEX 8.0, the method obtained results very close to optimal solutions in a short time on the four distributions in the CATS test sets. When applied to a new set of test cases described in [7], SAGII performs better than CPLEX, giving better result in much shorter time.

References

[1] A. Andersson, M. Tenhunen, and F. Ygge. Integer programming for combinatorial auction winner determination, 2000.

[2] Y. Fujishima, K. Leyton-Brown, and Y. Shoham. Taming the computational complexity of combinatorial auctions: Optimal and approximate approaches. In *Proceedings of International Joint Conference on Artificial Intelligence*, pages 548–553, 1999.

[3] Y. Guo, A. Lim, B. Rodrigues, and Y. Zhu. Heuristics for a brokering set packing problem. In *Proceedings of Eighth International Symposium on Artificial Intelligence and Mathematics*, 2004.

Test Instance	SAGII	t_1	CPLEX	δ_1	Casanova	t_2	δ_2
REL-1000-1000	86179.64	45.19	81755.45	5.13%	52048.73	113.55	39.60%
REL-1000-1000	83500.82	44.80	80479.80	3.62%	51340.27	111.16	38.51%
REL-1000-1000	85079.99	44.94	85079.99	0	54440.12	108.13	36.00%
REL-1000-1000	86747.23	44.58	81563.71	5.98%	54998.44	117.16	36.60%
REL-1000-1000	88513.37	44.58	83195.99	6.01%	51003.39	132.92	42.38%
REL-1000-1500	85101.43	71.05	79453.93	6.64%	53992.12	164.94	36.56%
REL-1000-1500	88086.29	67.56	74623.20	15.28%	58365.02	164.66	33.74%
REL-1000-1500	82046.16	69.03	80656.88	1.69%	58527.76	171.72	28.66%
REL-1000-1500	82341.22	68.11	78085.08	5.17%	55821.04	171.80	32.21%
REL-1000-1500	83772.91	67.09	76334.65	8.88%	56001.82	174.25	33.15%
REL-1500-1500	104346.07	90.73	92981.93	10.89%	65543.41	161.11	37.17%
REL-1500-1500	106056.08	90.95	98763.83	8.19%	64962.00	166.88	38.75%
REL-1500-1500	105699.93	91.09	98763.83	6.56%	70140.69	170.56	33.64%
REL-1500-1500	103252.95	90.42	92408.82	10.50%	65026.40	171.70	37.02%
REL-1500-1500	105462.71	91.22	94840.40	10.07%	62404.80	160.98	40.83%

Table 2. Experimental results on our new test set

Test set	# instance	t_1	μ_{SAGII}	t_2	$\mu_{Casanova}$	δ
REL-500-1000	100	38.06	64922.02	119.46	37053.78	42.93%
REL-1000-500	100	24.46	73922.10	57.74	51248.79	30.67%
REL-1000-1000	100	45.37	83728.34	111.42	51990.91	37.91%
REL-1000-1500	100	68.82	82651.49	168.24	56406.74	31.75%
REL-1500-1500	100	91.78	101739.64	165.92	65661.03	35.46%

Table 3. Comparison between Casanova and SAGII

[4] H. H. Hoos and C. Boutilier. Solving combinatorial auctions using stochastic local search. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence*, pages 22–29, 2000.

[5] B. Hudson and T. Sandholm. Effectiveness of preference elicitation in combinatorial auctions. In *Proceedings of AAMAS-02 workshop on Agent-Mediated Electronic Commerce (AMEC)*, pages 69–86, 2002.

[6] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 4598(13):671–680, May 1983.

[7] H. C. Lau and Y. G. Goh. An intelligent brokering system to support multi-agent web-based 4th-party logistics. In *Proceedings of the Fourteenth International Conference on Tools with Artificial Intelligence*, pages 10–11, 2002.

[8] K. Leyton-Brown, E. Nudelman, and Y. Shoham. Learning the empirical hardness of optimization problems: The case of combinatorial auctions. In *Eighth International Conference on Principles and Practice of Constraint Programming*, 2002.

[9] K. Leyton-Brown, M. Pearson, and Y. Shoham. Towards a universal test suite for combinatorial auction algorithms. In *ACM Conference on Electronic Commerce*, pages 66–76, 2000.

[10] T. Sandholm. Algorithm for optimal winner determination in combinatorial auctions. *Artificial Intelligence*, 135(1-2):1–54, 2002.

[11] T. Sandholm, S. Suri, A. Gilpin, and D. Levine. CABOB: A fast optimal algorithm for combinatorial auctions. In *Proceedings of Intenational Joint Conference on Artificial Intelligence*, pages 1102–1108, 2001.

[12] T. Sandholm, S. Suri, A. Gilpin, and D. Levine. Winner determination in combinatorial auction generalizations. In *Proceedings of AGENTS-2001 Workshop on Agent-Based Approaches to B2B, Montreal, Canada.*, 2001.

[13] D. Schuurmans, F. Southey, and R. C. Holte. The exponentiated subgradient algorithm for heuristic boolean programming. In *Proceedings of International Joint Conference on Artificial Intelligence*, pages 334–341, 2001.