Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computers & Operations Research 33 (2006) 2179–2188
computers & operations research
www.elsevier.com/locate/cor

Heuristics for a bidding problem

Y. Guo[a,*], A. Lim[b], B. Rodrigues[c], Y. Zhu[b]

[a]*Department of Computer Science, National University of Singapore, Science Drive 2, Singapore 117543, Singapore*
[b]*Department of IEEM, Hong Kong University of Science Technology, Clear Water Bay, Hong Kong*
[c]*School of Business, Singapore Management University, 469 Bukit Timah Road, Singapore 259756, Singapore*

Available online 2 February 2005

Abstract

In this paper, we study a bidding problem which can be modeled as a set packing problem. A simulated annealing heuristic with three local moves, including an embedded branch-and-bound move, is developed for the problem. We compared the heuristic with the CPLEX 8.0 solver and the current best non-exact method, Casanova, using the standard CATS benchmark and other realistic test sets. Results show that the heuristic outperforms CPLEX and Casanova.

Keywords: Bidding; Heuristics; Artificial intelligence

1. Introduction

In this work, we study a bidding problem which can be stated as follows: Assume there are n bids and m jobs where each bid can cover a number of jobs, and result in a profit to the supplier, w_j ($j \in 1, \ldots, n$), if bid j is selected. Take $[a_{ij}]_{m\times n}$ to be a m-row, n-column $0-1$ matrix, where $a_{ij} = 1$ if job i is included in bid j. Further, let $x_j = 1$ if bid j is selected and 0 otherwise. The integer program for the bidding problem is then given by

$$\text{maximize} \sum_{j\in\{1,\ldots,n\}} w_j x_j, \tag{1}$$

* Corresponding author.
E-mail addresses: guoyunso@comp.nus.edu.sg (Y. Guo), iealim@ust.hk (A. Lim), br@smu.edu.sg (B. Rodrigues).

0305-0548/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2005.01.007

$$\text{s.t.} \sum_{j\in\{1,\ldots,n\}} a_{ij}x_j \leqslant 1, \quad i \in \{1,\ldots,m\}. \tag{2}$$

This problem can be modeled as a set packing problem (SPP) which is well known to be **NP**-complete [1–5]. For a comprehensive survey of such problems and more general combinatorial auction problems, see de Vries and Vohra [6].

Exact algorithms, such as branch-and-bound [1] and iterative deepening A^* search [3] have been developed for the problem and applied to real-world data, and optimal solutions have been obtained using the ILOG CPLEX solver [2]. In addition, non-exact algorithms have been used, including an iterative greedy approach [5] and stochastic local search [4]. Because of the extensive practical application of the problem, mechanisms for generating realistic test suites have been studied and a standard benchmark test set, CATS, has been developed [7].

In [3], Sandholm developed a bid tree IDA* search, BOB, an early exact algorithm for the SPP where preprocessing techniques such as partitioning and removal of non-competitive bids were proposed. BOB is a depth-first branch-and-bound tree search on bids which uses nodes as bids and an edge between nodes to indicate conflict between the corresponding bids. However, BOB was not implemented by Sandholm. Yet another exact branch-and-bound algorithm for the SPP, called CASS, was proposed by Fujishima et al. [1] which they describe as a "naive brute-force approach followed by four improvements." In this work, a bids' bin concept, caching and a good bounding function were used and the algorithm performed well obtaining optimal solutions for test sets with about a hundred jobs and up to thousands of bids in reasonable times. CASS was faster than Sandholm's approach according to Anderson et al. [2] who proposed another exact algorithm which made use of CPLEX 6.5, which, although a general tool, achieved good results when compared with Sandholm's IDA* search and the CASS algorithm where solutions were found in shorter times for most test sets generated by CATS [2]. More recently, Sandholm et al. [8] proposed a second branch-and-bound-based algorithm called CABOB, which used techniques proposed in BOB. In fact, CABOB is, for most part, an implementation of BOB. Since in [2], CPLEX 6.5 was reported to have provided solutions faster than the IDA* algorithm, Sandholm et al. benchmarked CABOB against CPLEX 7.0, which, according to Sandholm et al., is 1.6 times faster than CPLEX 6.5 [8]. Experiments reported in this paper showed that CABOB was faster than CPLEX 7.0 for most CATS distributions used. We note that CPLEX 8.0 would now be a natural benchmark for the SPP since it is approximately 40% faster than CPLEX 7.0.

Besides exact algorithms, heuristics have been applied to the SPP. In [5], a non-complete greedy heuristic is applied to the SPP and in [4], a stochastic local search algorithm, Casanova, was reported to do better than CASS for test sets from [1] and [3] when cut off times were set for both CASS and Casanova. It outperformed CASS on large problems instances with other various distributions and generally found optimal solutions on smaller instances. The Casanova heuristic is the current best non-exact algorithm for the SPP we found in the literature and can be described briefly as follows: Begin with an empty allocation where all jobs are assigned to a dummy bid and any real bid is unsatisfied and perform a fixed number of local search steps: with a certain probability p, select any unsatisfied bid to include in the current solution and remove any conflicting bids originally in the solution; with probability $1-p$ select a bid greedily by ranking all bids not in solution according to their bid profit divided by the number of jobs covered in the bid in decreasing order. Then, let b_1 and b_2 be the first and second ranked bid. If b_2 has been inserted into the solution more recently than b_1, include b_1 in the current solution; otherwise with a probability

q include b_1, or with probability $1-q$ select b_2, after that remove all conflicting bids in the solution. The best solution obtained in the process is taken as solution [4].

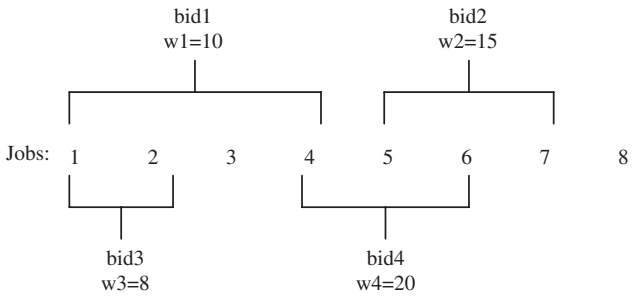
In this paper, we develop a new non-exact algorithm based on simulated annealing (SA) and hybrid neighborhood search techniques which consist of a branch-and-bound search, greedy selection and randomized 1, 2-exchanges. Preliminary results reported in [9] which outperformed the iterative greedy search proposed in [5] are improved, and the new algorithm is compared with CPLEX 8.0 and Casanova using test sets from CATS as well as more realistic data sets recently provided in [5].

2. Using simulated annealing with hybrid local moves

In the approach proposed here, SA [10] is used and local moves are applied to solve the SPP. The framework is given in Algorithm 1, where the constants p_1 and p_2 are set to be 0.2 and 0.7, respectively, which is the proportion of time allocated for the branch-and-bound and greedy moves, respectively. In the algorithm, preprocessing (using PB-1-1, PB-1-2, PB-2-1 explained in Section 2.1) is only executed once in the algorithm.

Algorithm 1 A Simulated Annealing Framework

```
S ← {}; best_value ← 0;
Temperature ← T_max; Iter ← 0
Preprocess_on_Bids()
while Iter < Max_Iter and
Temperature > T_Terminate do
  with probability p_1
    Stemp ← Branch_and_Bound_Move(S);
  with probability p_2
    Stemp ← Greedy_Move(S);
  with probability 1 − p_1 − p_2
    Stemp ← 1, 2-exchange(S);
  δ = value(Stemp) − value(S)
  if δ ⩾ 0 then
    S ← Stemp
  else {from S to Stemp is a downhill move}
    p = e^(δ/Temperature)
    with probability p
    S ← Stemp
  end if
  if value(S) > best_value then
    best_value ← value(S);
  end if
  iter ← iter + 1
  Temperature ← Temperature ∗ C_0
end while
```



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

Fig. 1. Example of PB-2-2.

2.1. Preprocessing bids

To improve on time, preprocessing attempts to exclude bids that can lead to suboptimal results. For example, when bids i and j cover the same set of jobs but have weights $w_i \geqslant w_j$, bid j will not be included in the solution. Here, these criteria are extended using the decision problem PB-n_1-n_2-J-$(b_1, \ldots, b_n)$: Given a set of jobs J, n bids $b_1, \ldots, b_n$, each being a subset of J of covered jobs, and positive integers n_1, n_2, with $n_1, n_2 \leqslant n$, can we find two sets of bids S_1 and S_2, such that

- $|S_1| = n_1$ and $|S_2| = n_2$,
- for $i, j \in \{1, \ldots, n\}$, $i \neq j$, if $b_i \in S_1$ and $b_j \in S_1$ then $b_i \bigcap b_j = \emptyset$,
- for $i, j \in \{1, \ldots, n\}$, $i \neq j$, if $b_i \in S_2$ and $b_j \in S_2$ then $b_i \bigcap b_j = \emptyset$,
- $\bigcup_{b_i \in S_1} b_i \subset \bigcup_{b_j \in S_2} b_j$,
- $\sum_{b_i \in S_1} w_i \geqslant \sum_{b_j \in S_2} w_j$?

If S_1 and S_2 can be found, S_2 is not included in the solution since the objective function value can be improved or maintained by replacing all bids in S_2 by all bids in S_1. This substitution is feasible since jobs covered by bids in S_1 are covered by bids in S_2, and all bids in S_1 are mutually exclusive. Fig. 1 provides an example.

PB-n_1-n_2-J-$(b_1, \ldots, b_n)$ is, however, **NP**-complete. To show this, we state the weighted SPP as a decision problem: Given a set S, a collection of subsets C of S, each with a weight $w(c)$, $c \in C$, and constant P, there is a collection of subsets $\overline{C} \subset C$ such that for any $c, c' \in \overline{C}$, $c \cap c' = \phi$ and $\sum_{c \in \overline{C}} w(c) \geqslant P$?

To establish PB-n_1-n_2-J-$(b_1, \ldots, b_n)$ is **NP**-complete, we need now only consider one bid $b = J$ (the entire job set) and set $w(b)$ to be P. It then follows by taking $n_2 = 1$ and $S_2 = \{b\}$ that PB-n_1-n_2-J-$(b_1, \ldots, b_n)$ has a solution if and only if the answer to the SPP decision problem above is "yes".

In the remainder of this paper, we shorten the notation PB-n_1-n_2-J-$(b_1, \ldots, b_n)$ with PB-n_1-n_2 when J and $(b_1, \ldots, b_n)$ are understood in context.

In the implementation of the algorithm, preprocessing is carried out only with PB-1-1, PB-1-2 and PB-2-1 since using PB-n_1-n_2 for other n_1, n_2 values is computationally prohibitive. In using these to eliminate sets of bids, the single bid S_2 obtained in PB-1-1 or PB-2-1 is removed. For, PB-1-2, the two bids in S_2 are not considered if the bid in S_1 is selected; however, if bids in S_1 are not selected, there is

still a possibility of selecting bids in S_2. Hence, bids in S_2 are not removed from consideration as is done for PB-1-1 and PB-2-1. Besides, the selection of bids in S_2 depends on whether these are in the same connected component or not. This is discussed further in Section 2.2.2 where implementation details are provided.

2.2. Local move 1: branch-and-bound

In [1], the branch-and-bound algorithm CASS was applied to the entire solution space which, given enough time, will find optimal solutions for any test set. CASS branches on the number of jobs and hence performs well on test sets which have small numbers of jobs (30 to 150) and relatively large number of bids (several thousands). The CPLEX IP solver exhibits better performance than CASS for most of the test sets generated from CATS, which usually contain several hundred jobs. In general, from experimentation, we found that a branch-and-bound search on the entire solution space for the SPP is less efficient than when branch-and-cut is used in CPLEX.

Here, branch-and-bound is employed as a local move by applying it to a fragment of the solution. In this approach, a subset of jobs and bids (a fragment) is selected and the optimal solution is sought while keeping other jobs and bids in the current solution fixed. The details are provided in the following sections:

2.2.1. Overview

If the current solution is a set of selected bids S, with $|S| = m_0$, the branch-and-bound local search randomly picks m_1 ($\leqslant m_0$) bids ($b_1, \ldots, b_{m_1}$) from S and removes them from the solution. Taking $\bigcup_{i=1}^{m_1} b_i$ to be K, and P be the maximum set of bids $b'_1, \ldots, b'_{|P|} \subset K$, branch-and-bound is used to find the optimal solution for bids in P. Once this is achieved, the solution inserted back into S. The solution thus obtained is feasible since bids in P do not conflict with bids in S. Also, the new solution is at least as good as the previous one since the partial optimal solution for P is found by branch-and-bound. In the implementation, small m_1 values were used so that the number of bids in P is small which results in faster searches.

2.2.2. Implementation

Preprocessing is used to improve the time efficiency. P, as above, is represented by a graph $G(V, E)$, in which each node is a bid and $|V| = |P|$. The presence of an edge in E from node i to node j indicates that bid i and bid j are in conflict. G is partitioned into its connected components so that the sum of optimal solutions of each connected component is the optimal solution for the bid set P. In this way, the search complexity of the problem can be reduced significantly.

In using PB-1-2, an edge is added between two bids if these are in the same connected component to ensure they are not selected together; on the other hand, if the bids are in different connected components, this is not done since, otherwise, the resulting larger connected component will increase the search space. Instead, bids in S_2 found using PB-1-2 are recorded and checked after the heuristic is applied. If any pair of these recorded bids is selected, they are substituted with the bid from S_1, which improves the solution without violating the constraints. Because of this, the potential from using PB-1-2 is fully realized.

The bounding function used is similar to the one used in the CASS algorithm. For each job i, we estimate the profit it can bring by $\pi(i) = \max(w_j/|b_j| : 1 \leqslant j \leqslant m, \text{job } i \in b_j)$. If the current solution S

consists of m_0 bids $(b_1, \ldots, b_{m_0})$, the bounding function is taken to be given by

$$bound(S) = value(S) + \sum_i \pi(i),$$

where $value(S)$ is the sum of profits of the m_0 bids in S, and the sum is taken over all i for which job i is covered by a bid in $S - P$.

Whenever this bound is smaller than the current best solution, the current branch is discarded. The branch-and-bound method used in a partial solution is an example of how we hybridized an exact search method within a non-exact heuristic search.

2.3. Local move 2: greedy local search

A greedy local search is used to find unselected bids which do not conflict with bids already in the current solution and have larger greedy values. Although bid profit can be used for the greedy value, a better and more refined value can be employed since the more a bid is in conflict with other bids, the more constraints there will be on the bids that can be chosen if it is included in the solution. Hence, the greedy value of each bid b_i was taken to be the profit of b_i offset by a penalty for b_i, which is given by

$$penalty_i = \sum_{j=1}^{n} \left[C_1 \cdot w_j - C_2 \cdot \sum_{k=1}^{n} w_k \cdot c_{jk} \right] \cdot c_{ij},$$

where $[c_{ij}]_{n\times n}$ is a n-row, n-column 0-1 matrix where $c_{ij} = 1$ if and only if there is a $k \in \{1, \ldots, n\}$ with $a_{ik} = 1$ and $a_{jk} = 1$, and C_1 and C_2 are scaling factors. Here, each bid in conflict with b_i contributes its scaled profit reduced by a scaled sum of profits of bids it conflicts with. This value allows us to penalize b_i by bids it is in conflict with. Since each of these conflicting bids can, in turn, be in conflict with other bids, we offset the penalty with the sum of profits from the latter set of bids. This would discourage moves to solutions which include bids that conflict with many other bids and therefore help generate fewer conflicts. After tuning, C_1 and C_2 were set to be approximately 10^{-n} and 10^{-2n}, respectively, when the number of bids was of magnitude 10^n.

2.4. Local move 3: 1, 2-exchange

The 1- and 2-exchange moves are random in the solution space. These randomly pick 1 or 2 unselected bids from the candidate bid set. For a 2-exchange move, 2 bids are selected which do not conflict with each other and any bid in the current solution that conflicts with the newly selected bids is removed and the new bids added to the current solution.

3. Computational experiments

The algorithm, SAGII, was compared with the current best exact and non-exact algorithms for the SPP, namely CPLEX 8.0 and the Casanova algorithm, respectively. CPLEX was implemented using the SPP model provided in [2]. As we were unable to secure the Casanova program from the authors [4], we implemented Casanova according to the authors description and tuned all parameters extensively.

Table 1
Results using CATS benchmark sets

Test set	# of instances	t_1	μ_{CPLEX}	t_2	μ_{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%

The three algorithms were first compared with test sets generated by CATS used in [2] with four different distributions. Test sets from [5] of various sizes were also used. These were different from CATS sets since real-world factors were incorporated. All experiments were run on Pentium 4 2.40 GHz machines with 1 Gb of memory.

3.1. Comparisons using CATS test sets

In [7], Leyton-Brown et al. discussed the need to have a standard test suite for combinatorial auction problems, and discussed generating test suites mainly by varying distributions. CATS programs were developed for researchers to generate their own test sets with different problem sizes and distributions. There have been a number of studies which used CATS as a standard to measure performance: see, for example, [11–13]. In this work, test sets from [2] were used. In these sets, for example, a test set with a uniform distribution, p jobs and q bids is denoted by UNI-p-q. Here, four categories of test sets were used with random, binomial, exponential and uniform distributions to test the performance of CPLEX, Casanova and SAGII. The experimental results are given in Table 1 where t_1, t_2 and t_3 are the average times in seconds. The 10 instances of each category required using CPLEX, Casanova and SAGII, respectively, and μ is the average value of results obtained for the 10 test sets. The values δ_1 and δ_2 are the differences in percentages of the Casanova and SAGII methods, respectively, from optimal solutions obtained by CPLEX.

From the table, we see that SAGII consistently outperforms Casanova providing better results for all distributions within shorter times and with significantly smaller deviations. We note that in Casanova, two types of local searches are used: a totally random move and a greedy local move. In SAGII, however, in addition to random local moves, i.e. the 1, 2-exchanges, guided local searches were used, namely a branch-and-bound search and a greedy local search. These, together with the simulated annealing heuristic, provided the improved solutions.

When compared with CPLEX, SAGII gave solutions consistently within 5% from the optimal for the random, binomial and exponential distributed sets and achieved optimal solutions for all test sets with the binomial distribution. The time required was less than 1/6 of the time required by CPLEX. For the uniform distribution, CPLEX was about 15% better than SAGII while Casanova solutions were more than 40% from optimal values.

In the experiments, CPLEX was found to be the best exact algorithm for CATS test sets. When analyzing log reports from CPLEX, it was found that CATS-generated test sets were easily solved with CPLEX as the number of Gomory fractional cuts made was usually very small. In [8], CATS sets were reported to be easily solved by CPLEX and CABOB.

Table 2
Results using more realistic test sets

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%

3.2. Comparisons using more realistic test sets

For further comparisons, the performance of CPLEX, Casanova and SAGII was tested using new data provided by Lau and Goh [5], where more test sets with comparable number of jobs and bids were used consisting of up to 1500 jobs and 1500 bids. These test sets allow for several factors including a pricing factor which models a bidder's acceptable price range for each bid, a preference factor which takes into account bidder's preferences among bids, and a fairness factor which measures the fairness in distributing jobs among bidders. Detailed explanations of how these attributes are incorporated into test data can be found in [5]. Test sets are labeled REL-*n*-*m*, where *n* is the number of jobs and *m* is the number of bids. For these, since CPLEX was unable to obtain optimal solutions within reasonable times, it was given the advantage by allowing it 3600 s for each of the 15 test cases. Results are given in Table 2 where t_1 and t_2 is the time required for SAGII and Casanova, respectively, for each test instance, and δ_1 is the percentage difference of SAGII over CPLEX, and δ_2 is the percentage difference of CPLEX over Casanova.

From the table, it can be seen that SAGII consistently outperformed CPLEX achieving better results in much shorter times for every instance. The new algorithm was 4.15% better than CPLEX for the REL-1000-1000 test set, 7.53% better for the REL-1000-1500 test set and 9.24% better for the REL-1500-1500 test set. Further, SAGII used approximately 1/80 to 1/40 the time required by CPLEX on these test sets and improved as size increased.

When comparing SAGII with Casanova, we can see that SAGII outperformed the Casanova achieving more than 30% better margins in about half the time required by Casanova.

3.3. Further experimentation with SAGII and Casanova

The algorithms were compared further using 500 test instances generated in [5] with 5 different problem sizes. Table 3 provides the results, where t_1 and t_2 are the average times in seconds required for SAGII

Table 3
SAGII vs. Casanova

Test set	# instance	t_1	μ_{SAGII}	t_2	μ_{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%

and Casanova, respectively, μ_{SAGII} and μ_{Casanova} are the average values of the solutions obtained by the algorithms, and δ is $(\mu_{\text{SAGII}} - \mu_{\text{Casanova}})/\mu_{\text{SAGII}}$.

When applied to these test sets, Casanova failed to obtain the same quality of results as SAGII which always outperformed Casanova with a margin of more than 30% while requiring only about half the running time.

4. Conclusion

In this paper, we studied a bidding problem formulated as a set packing problem (SPP) and surveyed methods available in the literature where exact and non-exact algorithms have been developed for the problem. A heuristic using simulated annealing with preprocessing and three local moves was proposed. When compared with CPLEX 8.0, the new algorithm achieved comparable results in shorter times on the CATS benchmarks tests, and significantly better solutions for more realistic test sets developed recently. Results from extensive computational experiments also showed that the new heuristic outperforms the current best heuristic for the SPP, Casanova.

In future work, the relationships between test data structure and the efficacy of the various exact and heuristic approaches available could be studied.

References

[1] Fujishima Y, Leyton-Brown K, Shoham Y. Taming the computational complexity of combinatorial auctions: optimal and approximate approaches. Sixteenth international joint conference on artificial intelligence, 1999, p. 548–53.
[2] Andersson A, Tenhunen M, Ygge F. Integer programming for combinatorial auction winner determination. Proceedings of fourth international conference on multiagent systems, 2000, p. 39–46.
[3] Sandholm T. Algorithms for optimal winner determination in combinatorial auctions. Artif Intell 1999;135(1–2):1–54.
[4] Hoos H, Boutilier C. Solving combinatorial auctions using stochastic local search. Proceedings of the 17th national conference on artificial intelligence, 2000, p. 22–9.
[5] Lau HC, Goh, YG. An intelligent brokering system to support multi-agent web-based 4th-party logistics. Proceedings of the 14th international conference on tools with artificial intelligence, 2002, p. 154–61.
[6] de Vries S, Vohra R. Combinatorial auctions: a survey. INFORMS J Comput 2003;15:284–309.
[7] Leyton-Brown K, Pearson M, Shoham Y. Towards a universal test suite for combinatorial auction algorithms. ACM conference on electronic commerce, 2000, p. 66–76.
[8] Sandholm T, Suri S, Gilpin A, Levine D. CABOB: a fast optimal algorithm for combinatorial auctions. International joint conferences on artificial intelligence, 2001, p. 1102–8.

[9] Guo Y, Lim A, Rodrigues B, Zhu Y. Heuristics for a brokering set packing problem. Proceedings of eighth international symposium on artificial intelligence and mathematics, 2004, p. 10–14.
[10] Kirkpatrick S, Gelatt CD, Vecchi MP. Optimization by simulated annealing. Science 1983;220:671–80.
[11] Hudson B, Sandholm T. Effectiveness of preference elicitation in combinatorial auctions. AAMAS-02 workshop on agent-mediated electronic commerce (AMEC), 2002, p. 69–86.
[12] Schuurmans D, Southey F, Holte RC. The exponentiated subgradient algorithm for heuristic boolean programming. International joint conference on artificial intelligence, 2001, p. 334–41.
[13] Sandholm T, Suri S, Gilpin A, Levine D. Winner determination in combinatorial auction generalizations, 2001. AGENTS-2001 workshop on agent-based approaches to B2B, Montreal, Canada.