Soft Comput (2009) 13:905–917
DOI 10.1007/s00500-008-0355-3

FOCUS

A memetic algorithm for the optimal winner determination problem

Dalila Boughaci · Belaïd Benhamou · Habiba Drias

Published online: 26 July 2008

Abstract In this paper, we propose a memetic algorithm for the optimal winner determination problem in combinatorial auctions. First, we investigate a new selection strategy based on both fitness and diversity to choose individuals to participate in the reproduction phase of the memetic algorithm. The resulting algorithm is enhanced by using a stochastic local search (SLS) component combined with a specific crossover operator. This operator is used to identify promising search regions while the stochastic local search performs an intensified search of solutions around these regions. Experiments on various realistic instances of the considered problem are performed to show and compare the effectiveness of our approach.

1 Introduction

An auction is a market protocol for the coordination of agent activities or for resource allocation in multi-agent systems. The combinatorial auction (CA) is the mechanism that allows agents (bidders) to bid on bundles of items (goods). It allows the bidders to express both complementarity[1] and substitutability[2] of their preferences within bids. The combinatorial auction avoids the risk to obtain incomplete bundles since the seller allows bids on bundles of items.

Combinatorial auctions have been used in various domains such as economics, game theory and task allocation in multi-agent systems (Rothkopf et al. 1998; Fujishima et al. 1999; Leyton-Brown et al. 2000b; Boughaci and Drias 2005; Collins et al. 2000). They have been also used in real-world applications such as the sale of spectrum licenses in America's Federal Communications Commissions (FCC)[3] auctions.

In this work, we are interested in the optimal winner determination problem (WDP) in combinatorial auctions. The WDP is a complex problem, it is equivalent to the weighted set packing problem which is *NP-Complete* (Rothkopf et al. 1998). Given a set of bundles bids, the winner determination problem is to decide which of the bids to accept. The accepted bids must ensure optimality and feasibility. The problem is expressed as follows:

1.1 The problem formalization

The optimal winner determination problem in combinatorial auctions can be stated as follows:

Let us consider a set of m items, $M = \{1, 2, \ldots, m\}$ to be auctioned and a set of n bids, $B = \{B_1, B_2, \ldots, B_n\}$. A bid B_j is a tuple $\prec S_j, P_j \succ$ where S_j is a set of items, and P_j is

D. Boughaci (✉) · B. Benhamou
INCA/LSIS, CMI 39 rue Fredric Joliot-Curie,
13013 Marseille, France
e-mail: boughaci@cmi.univ-mrs.fr

B. Benhamou
e-mail: belaid.benhamou@cmi.univ-mrs.fr

D. Boughaci · H. Drias
LRIA/USTHB, BP 32 El-Alia, Beb-Ezzoaur,
16111 Algiers, Algeria
e-mail: dboughaci@usthb.dz

H. Drias
e-mail: hdrias@usthb.dz

[1] Complementarity between items means that the value assigned to a collection of goods is greater than the sum of the values assigned to its individual's elements.

[2] Substitutability means that the value assigned to a collection of goods is lower than the sum of the value attached to its individual's elements.

[3] http://wireless.fcc.gov/auctions.

the global price of the items of S_j ($P_j \geq 0$). Further, consider a matrix $a_{m\times n}$ having m rows and n columns where $a_{ij} = 1$ iff the item i belongs to S_j, $a_{ij} = 0$, otherwise. Finally the decision variables are defined as follows: $x_j = 1$ iff the bid B_j is accepted (a winning bid), and $x_j = 0$ otherwise (a losing bid).

The WDP is the problem of finding winning bids that maximize the auctioneer's revenue under the constraint that each item can be allocated to at most one bidder. The WDP can be modeled as the following integer program (Sandholm 1999):

$$\textbf{Maximize} \quad \sum_{j=1}^{n} P_j.x_j \tag{1}$$

$$\textbf{Under the constraints}: \sum_{j=1}^{n} a_{ij}x_j \leq 1 \qquad i \in \{1, \ldots, m\} \tag{2}$$

$$x_j \in \{0, 1\} \tag{3}$$

The objective function (1) maximizes the auctioneer's revenue which is equal to the sum of the prices of the winning bids. The constraints (2) express the fact that each item can be allocated to at most one bidder. Due to the free disposal assumption, some items could be left uncovered. Now we address the methodology that we propose to solve the WDP problem.

1.2 The proposed methodology

Classical optimization methods such as A^*, for example, encounters a great difficulty when faced with the challenge of solving hard problems that bound in the real world. Metaheuristics go beyond the classical techniques to provide methods that are able to solve problems of practical significance in reasonable time. In this work, we are interested in a memetic algorithm for the WDP. The memetic algorithm (MA) Moscato (1989) is an evolutionary metaheuristic like a genetic algorithm. The MA approach is inspired by the Darwinian principles on natural evolution and the notion of a meme of Dawkins, which is defined as a cultural evolution unit that performs local refinements (Dawkins 1976). The MA can be viewed as a genetic algorithm combined with some kinds of local search. Recent studies have shown that memetic approaches can lead to high quality solutions more efficiently than genetic algorithms (Hart William et al. 2005; Ong et al. 2007). MA has been successfully applied on many complex problems (Boughaci et al. 2004; Burke et al. 2001; Caponio et al. 2007; Franca et al. 2001; Ishibuchi and Narukawa 2004; Ishibuchi et al. 2003; Neri et al. 2007; Ong et al. 2006; Tang et al. 2007; Tang and Yao 2007; Zhou et al. 2007). The ability of MA in solving large problems motivates our choice to use memetic concepts to solve the WDP.

In this paper, we develop a new memetic algorithm for the WDP. The main contributions of this work are summarized in the following points:

1. The first point consists in generating individuals, the proposed algorithm makes use of the random key encoding (RK) introduced by Bean (1994) mainly to solve ordering and scheduling problems. The RK encoding mechanism allows to generate and manipulate feasible individuals, then avoids additional penalties for invalid individuals.
2. Secondly, we propose a new selection strategy based on both fitness and diversity to choose a list of candidate individuals in order to participate in the reproduction phase and generate other individuals.
3. The third point consists in enhancing the proposed strategy by using a specific crossover operator combined with a SLS as an improvement technique. Our objective is to achieve a good compromise between intensification and diversification in the search process.
4. The fourth point consists on the way to manage the population. Our approach takes into account the diversity and the fitness criteria to add a new individual to the current collection. That is, the new individual is added to the current collection when it improves either the quality or the diversity of the collection. Otherwise, it is discarded.
5. Finally, to maintain a feasible allocation along the memetic search process, we have defined a conflict graph where the vertices are the bids and the edges connect bids that cannot be accepted together. This graph is used to remove any conflicting bids[4] occurring in the current allocation when new bids are added.

The rest of the paper is organized as follows. Section 2 describes the new memetic algorithm that we propose for the WDP problem. Some experimental results are reported in Sect. 3. Section 4 presents some previous works on the WDP problem. Finally, Sect. 5 concludes the work.

2 The new memetic algorithm for the WDP problem

2.1 The principle

The memetic algorithm that we propose here starts with an initial population P of individuals[5] created randomly according to the random key encoding (RK). It then selects a collection C of individuals of size $|C|$ [6] from the current population in order to participate in the reproduction phase.

[4] Conflicting bids are those sharing goods.

[5] Individuals represent solution candidates for the WDP problem.

[6] $|C|$ is the cardinality of the collection C.

The collection C contains, on one hand, $|C_1|$ highest-fitness individuals, that are selected from the population P according to their fitness value. On the other hand, $|C_2|$ other individuals from $P - C_1$ are added to the collection C to complete it. The individuals of C_2 are called diverse individuals since they are the most distant from the individuals in C. The diversity of an individual is measured by a similarity function that computes the number of the same genes existing between two individuals. The novel strategy of selection helps the algorithm to maintain at each generation a good and diversified population which lead to a good compromise between intensification and diversification. The size of the collection C, $|C| = |C_1| + |C_2|$, is fixed by an empirical study.

After selecting a set of good and diverse individuals, the reproduction phase starts. Once two parents have been selected, their chromosomes are combined and a new individual is generated. In order to locate solutions more effectively, the mutation phase is replaced by a stochastic local search.

Unlike a classical genetic algorithm that replaces the bad individuals by the fittest new ones, the memetic approach inserts the new individual in the current collection of individuals according to both their fitness and diversity values. A new individual is added to the best solutions of C_1 and the worst one is removed when the new individual improves the quality of the current collection C. Otherwise, if the new individual improves the diversity of the current collection C, then the individual in the collection having a big similarity value is replaced by the new one.

2.2 The memetic algorithm components

The components of the memetic algorithm for solving the WDP problem are defined as follows:

2.2.1 The individual representation

The individual representation is defined as a combination of bids satisfying the target goals described in the objective function. We use an integer vector A having a variable length bounded by the number of bids n where each component A_i receives the number of a winning bid. But when there is no confusing, we consider in our explanation as components of the vector A the winning bids themselves.

An individual is generated randomly according to the Random Key Encoding that operates as follows: we generate n real numbers sequenced by an r order, where n is the number of bids and the r order is a permutation of key values. To generate an allocation, first we select the bid having the highest order value and include it in the allocation. Secondly, the bid having the second-highest order value is accepted if it does not conflict with bids that are already in the allocation, otherwise it is discarded. The process continues until having examined the n bids. We obtain a subset of bids that may be a feasible solution to the WDP.

Consider for instance a set of four items $\{1, 2, 3, 4\}$ to be auctioned and three bids $\{B_1, B_2, B_3\}$. Each bid specifies the price P_i that the bidder proposed to pay for a particular bundle S_i. Suppose that the proposed bids are the following:

$$Bid\ 1 : B_1 = (\{1, 2\}, 500.25)$$
$$Bid\ 2 : B_2 = (\{1, 2, 3\}, 678)$$
$$Bid\ 3 : B_3 = (\{3, 4\}, 200.10)$$

To generate a feasible individual for this example, we follow these steps:

1. First, generate three random real numbers, for example, $r = (0.65, 0.70, 0.80)$. The first bid to accept is the bid B_3 because it has the highest order value 0.80. The current allocation receives the bid B_3. $A = (B_3)$.
2. The bid having the second-highest order value is B_2, but it is discarded because it conflicts with the bid B_3[7] currently in A.
3. Finally, the bid B_1 can be added to the allocation A because it does not conflict with the bids in A.

We obtain the allocation $A = (B_3, B_1)$ that can be one of the solutions for the WDP. The overall price is simply the sum of prices of the winning bids $=200.10 + 500.25 = 700.35$. From the allocation A, we can say that the winning bids are B_3 and B_1 and B_2 is a losing bid.

2.2.2 The objective function

The objective process consists in finding a solution that maximizes the bidder revenue. On other words we need to compute a feasible allocation A that maximizes the objective function. The objective function represent the fitness of the collection A which is simply expressed by the overall price of the winning bids of A. Formally, it is encoded by the formula: $fitness(A) = \sum_{i=1}^{L} Price(B_i)$ where L is the number of bids in the allocation A.

2.2.3 The diversity/similarity function

This function computes the number of the same genes between two individuals which represents their similarity. It is used to choose a collection of diverse individuals that will participate in the reproduction phase. It is clear that an individual that has a small similarity value to the other individuals that are already in the current collection will contribute to the diversity of the collection.

[7] The bids B_3 and B_2 shares the same item 3.

Given the similarity measure $Sm(X, Y)$ between two individuals X and Y, the similarity value $Sm_C(X)$ of an individual X to a set of individuals C is given by the formula: $Sm_C(X) = \max_{Y \in C}(Sm(X, Y))$

Example 1 We illustrate in the following how to compute the similarity value of individuals.

1. The similarity value between $X = (\mathbf{B_1}, B_2, \mathbf{B_3})$ and $Y = (\mathbf{B_3}, \mathbf{B_1}, B_4, B_5)$ is equal to 2.
2. The similarity value between $X = (\mathbf{B_1}, \mathbf{B_2}, \mathbf{B_3})$ and $Y = (\mathbf{B_3}, \mathbf{B_2}, B_5, \mathbf{B_1})$ is equal to 3.
3. The similarity value between $X = (B_1, B_2, B_3)$ and the set of the two individuals $Y = (B_3, B_1, B_4, B_5)$ and $Z = (B_3, B_2, B_5, B_1)$ is equal to 3. It is equal to the maximum between $Sm(X, Y)$ and $Sm(X, Z)$ values.

The diversity function is based on similarity comparison. To compute the diversity value of a solution X to the collection C requires calculating $|C|$ diversity values which is computationally expensive. This is one of the reasons why the new selection strategy is more effective when applied to a small collection of individuals.

2.2.4 The collection selection strategy

The selection strategy that we propose focuses on both diverse and quality criteria to choose individuals. The individuals represent solution candidates to be improved, and from which other individuals are generated. These individuals are selected as follows:

1. First, we select from the current population P a set of $|C_1|$ best individuals. The collection C receives initially the C_1 highest-fitness individuals.
2. Secondly, we compute for each individual V from the rest of the population $P - C_1$, its similarity value to the current collection C. To ensure high level of diversity in the collection, we select from $P - C_1$, a number of $|C_2|$ diverse individuals having small similarity values to complete the collection C.

Finally, we obtain a collection C of C_1 highest-fitness individuals and C_2 diverse individuals.

2.2.5 The crossover operator

The crossover operator that we propose takes into consideration the semantic of the individuals to obtain efficient children. It takes two individuals (called parents) and produces a new individual (called a child). From the first parent to the end of the second parent, the operator decides which parent will give its gene value to the child; and all conflicting bids are discarded as shown in Algorithm 1.

Consider for example a set of five items $M = \{1, 2, 3, 4, 5\}$ to be auctioned and five bids $B = \{B_1, B_2, B_3, B_4, B_5\}$. Each bid specifies which price P_i that the bidder propose to pay for a particular bundle of items. Suppose that the proposed bids are the following:

$Bid\ 1 : B_1 = (\{1, 2\}, 500)$
$Bid\ 2 : B_2 = (\{2, 3\}, 200)$
$Bid\ 3 : B_3 = (\{3, 5\}, 300)$
$Bid\ 4 : B_4 = (\{4\}, 100)$
$Bid\ 5 : B_5 = (\{1\}, 100)$

Now, let us consider two allocation parents X and Y such that $X = (B_1, B_4)$ and $Y = (B_3, B_5, B_4)$ having the revenues 600 and 500, respectively. To construct a child Z, we apply the crossover operator that operates as follows:

To build the child Z, we take from the parent X, the bids B_1 and B_4, and from the parent Y the bid B_3. The bid B_5 of Y conflicts with the bid B_1 of Z since they share the same item 1, then it is discarded.

Finally, We obtain the child $Z = (B_1, B_4, B_3)$ having a revenue $= 500+100+300 = 900$. The code of the crossover operator is given in Algorithm 1.

Algorithm 1 : The crossover operator.

Require: two parents $Parent1$ and $Parent2$
Ensure: A child, $Child$
1: $Child \Leftarrow \phi$
2: **for** each gene from the beginning of $Parent1$ to the end of the $Parent2$ **do**
3: **if** (there is no conflict) **then**
4: $Child \Leftarrow Child$ with a *gene* value included into it
5: **end if**
6: **end for**
return the individual $Child$.

2.2.6 The stochastic local search method

In order to explore the most important part of the whole search space for attaining good solutions, we propose a stochastic local search (SLS) technique. The SLS technique starts with a generated individual V, then performs a certain number of local steps that consists in selecting a bid to be added in the individual V. The added bid is selected according to one of the two following criteria:

1. The first criterion (*step* 1 of Algorithm 2) consists in choosing the bid in a random way with a fixed probability $wp > 0$.

2. The second criterion (*step* 2 of Algorithm 2) consists in choosing the best bid. That is, the one maximizing the auctioneer's revenue.

The process is repeated until a certain number of iterations called $maxiter$ is reached. At each step, a new bid is included in the current allocation V and the conflicting bids of V are removed from it. The SLS method is sketched in Algorithm 2.

Algorithm 2 : The SLS improved method.

Require: a WDP instance, a Child V, maxiter, wp
Ensure: an improved individual V
1: **for** $I = 1$ to $maxiter$ **do**
2: r $\Leftarrow$ random number between 0 and 1
3: **if** r $\prec$ wp **then**
4: $bid =$ pick a random bid (*$Step\ 1$)
5: **else**
6: $bid =$ pick a best bid; (*$Step\ 2$)
7: **end if**
8: Add bid in the current solution V;
9: Remove all the conflicting bids from V ;
10: **end for**
return the best individual solution found.

The overall algorithm for the WDP is sketched in Algorithm 3.

Algorithm 3 : The Memetic Algorithm for the WDP.

Require: an instance of WDP.
Ensure: an allocation of bids that maximizes the auctioneer's revenue
1: Create the conflict graph
2: Generate randomly an initial population P according to the RK encoding
3: Select a list of candidate individuals C from P using the new selection strategy
4: **while** (the maximum number of generations is not reached) **do**
5: **repeat**
6: Select two individuals from C
7: Apply the crossover to obtain a new individual V
8: Apply the SLS on V
9: **if** (V improves the quality of C) **then**
10: Add V to the C_1 best individuals
11: Remove from C the worst one
12: **else if** (V improves the diversity of C) **then**
13: Add V to the C_2 diverse individuals
14: Remove from C the less diversified one
15: **end if**
16: **until** (All the parent combinations are examined)
17: **end while**
return the best individual solution found.

2.2.7 The conflict graph

To ensure feasibility of allocations during the memetic search process, we implemented a conflict graph having as a set of vertices the bids, and its edges connect the pairs of conflicting bids (bids sharing items).

3 Computational experiments

This section gives some experiment results. The source code is written in C language and run on a Pentium-IV 2.8 GHz, with 1GB of RAM.

To show the effectiveness of our approach, we compared the memetic algorithm (MA) with a genetic algorithm (GA) that we have implemented. The method GA uses a standard selection strategy, a same specific crossover operator as the one used by MA and a mutation operator without a local search. The selection strategy of GA is standard and consists in a fitness-based process and its mutation operator consists in selecting a random bid to be included in the current individual. A comparative study with some other algorithms of the state of the art concerning the WDP like Casanova (Hoos and Boutilier 2000) and SAGII (Guo et al. 2006) is also given in this section.

3.1 Benchmarks

To measure the performance of algorithms on the WDP problem, Leyton-Brown et al. developed the program combinatorial auction test suite (CATS) (Leyton-Brown et al. 2000a) to generate benchmarks. Recently Lau and Goh provided new data of various sizes consisting of up to 1500 items and 1500 bids (Lau and Goh 2002). These data sets allow for several factors such as a pricing factor, a bidder preference factor and a fairness factor in distributing items among bids. The CATS instances are easily solved by CPLEX and CABoB (Sandholm et al. 2001). In this paper, we use the realistic data pre-generated by Lau and Goh (2002) for which the CPLEX was unable to find the optimal solution in reasonable time (Guo et al. 2006). The pre-generated data set includes 500 instances and it is available at the Zhuyi's home page.[8] These instances can be divided into 5 different groups of problems where each group contains 100 instances.

If m is the number of items and n is the number of bids then the details of each group are given as follows:

- REL-1000-500 : 100 instances from in 101 to in 200: $m = 500, n = 1000$.
- REL-1000-1000 : 100 instances from in 201 to in 300: $m = 1000, n = 1000$.
- REL-500- 1000 :100 instances from in 401 to in 500: $m = 1000, n = 500$.
- REL-1500-1000 : 100 instances from in 501 to in 600: $m = 1000, n = 1500$.
- REL-1500-1500 : 100 instances from in 601 to in 700: $m = 1500, n = 1500$.

[8] (http://logistics.ust.hk/~zhuyi/instance.zip)

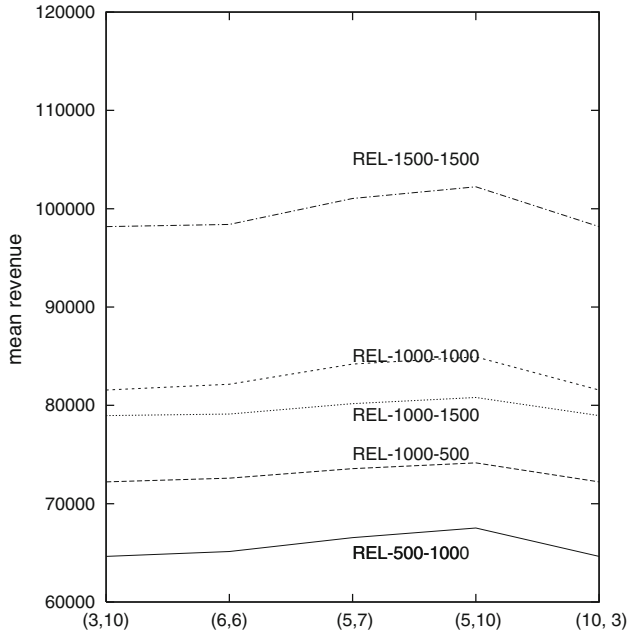

Fig. 1 The impact of $|C|$ on the solution quality of the method MA

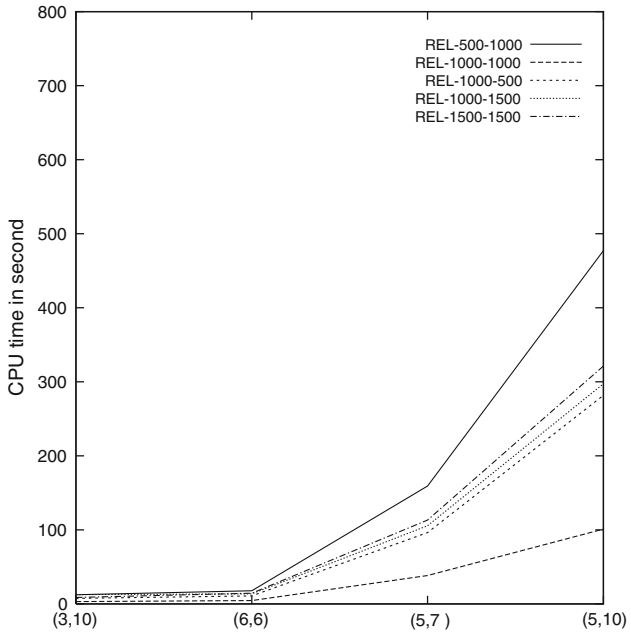

Fig. 2 The impact of $|C|$ on the efficiency of the MA

3.2 Parameters tuning

The adjustment of parameters of the MA algorithm is fixed by an experimental study. The MA parameters are: the population size ($popsize$), the size of the collection C ($|C| = |C_1|+|C_2|$), the maximum number of generations ($maxgen$). The local search improvement phase MA performs a number of iterations at each call equals to $maxiter$, and the probability wp is fixed empirically to 0.3.

3.2.1 The impact of the size of the collection C on the performance of MA

To examine the impact of $|C|$ on the efficiency of the method MA, several experiments have been done on the instances of the five different groups.

Figure 1 shows the impact of the parameter $|C|$ on the solution quality of MA. We can see that for each group of problems, the quality of solutions is improved when $|C|$ increases. Figure 2 shows that the CPU time of MA grows when $|C|$ increases.

3.2.2 The impact of maxiter on MA

Figure 3 (respectively Fig. 4) shows the impact of the maximum number of iterations ($maxiter$) of the SLS method on solution quality (respectively on CPU time) of the MA method.

We can see that in general both the solution quality and CPU time grow when $maxiter$ is increased.

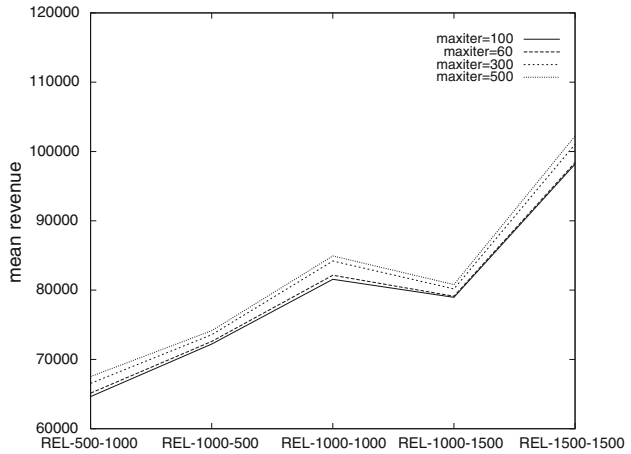

Fig. 3 The impact of $maxiter$ of SLS on the MA solution quality

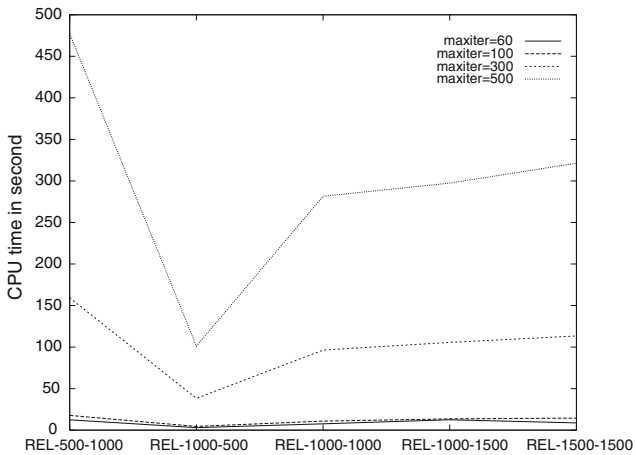

Fig. 4 The impact of $maxiter$ of SLS on the CPU time of MA

Table 1 The results of MA on instances of the WDP for different parameters

Test set	$\#ins$	Max Gen	pop Size	$\|C\|$	Max iter	Time	μ_{MA}
REL-500-1000	100	100	100	3,10	100	12.36	64644.93
REL-1000-500	100	100	100	3,10	100	3.07	72219.16
REL-1000-1000	100	100	100	3,10	100	7.57	81565.07
REL-1000-1500	100	100	100	3,10	100	12.49	78963.38
REL-1500-1500	100	100	100	3,10	100	8.69	98177.20
REL-500-1000	100	200	60	6,6	60	17.72	65133.21
REL-1000-500	100	200	60	6,6	60	4.48	72596.19
REL-1000-1000	100	200	60	6,6	60	10.87	82149.41
REL-1000-1500	100	200	60	6,6	60	13.56	79114.62
REL-1500-1500	100	200	60	6,6	60	14.52	98401.08
REL-500-1000	100	300	300	5,7	300	**159.30**	**66544.93**
REL-1000-500	100	300	300	5,7	300	**38.30**	**73562.89**
REL-1000-1000	100	300	300	5,7	300	**96.37**	**84199.99**
REL-1000-1500	100	300	300	5,7	300	**105.66**	**80173.42**
REL-1500-1500	100	300	300	5,7	300	**113.31**	**101035.52**
REL-500-1000	100	500	500	5,10	500	477.22	67520.23
REL-1000-500	100	500	500	5,10	500	101.12	74149.49
REL-1000-1000	100	500	500	5,10	500	281.63	84926.39
REL-1000-1500	100	500	500	5,10	500	297.35	80805.98
REL-1500-1500	100	500	500	5,10	500	321.27	102234.80

3.2.3 The effective sets of parameters

To further illustrate how the different MA parameters are adjusted, we give an overview of the experimental study performed for different parameters. Table 1 shows the results found MA on the five groups of instances where $\#ins$ corresponds to the number of instances, μ_{MA} corresponds to the average revenue found by MA and $time$ is the average CPU time of the algorithm given in second.

We can see from Table 1 that in general better solutions are found by MA When the values of the parameters increase, and the MA process requires more CPU time.

3.3 Experimental results

We carried several experiments to evaluate the performance of MA on the WDP and compared it to some other methods.

3.3.1 A comparison with the genetic algorithm (GA)

To show the improvement in the performance of MA when the new selection strategy is used with the SLS improvement technique, we compared it with GA on some instances of the WDP.

The parameters of GA are fixed by an empirical study as follows: $maxgen = 100$, $popsize = 25$, $crossover\ rate = 0.6$ and a $mutation\ rate = 0.1$. The fixed parameters values of MA are those for which a good compromise between solution quality and CPU time is obtained (see the bold lines of Table 1). They are set as follows: the collection C includes five higher quality individuals and seven diversified individuals, the population contains 300 individuals, the number of generations is fixed to 100 and the number of iterations of the local search is 300.

Tables 2, 3, 4, 5 and 6 depict the results of both MA and GA on several instances of the WDP problem where sol corresponds to the solution quality found by each algorithm and t to the CPU time in seconds of the algorithm.

Tables 2 and 3 give the results concerning some instances of the groups REL-1000-500 and REL-1000-1000 where we can see that MA outperforms GA in both the solution quality and CPU time.

Table 4 shows the results found by MA and GA on some instances from the group REL-500-1000. We can also see from this table that MA outperforms GA on all the checked

Table 2 MA versus GA on some of the REL-1000-500 instances

Instances	GA		MA	
	t	*sol*	*t*	*sol*
in101	336.90	42100.71	**129.62**	**67101.93**
in102	432.76	39641.22	**132.18**	**67797.61**
in103	338.89	43376.54	**133.34**	**66350.99**
in104	376.37	42790.65	**135.14**	**64618.41**
in105	331.31	40841.21	**153.96**	**66376.83**
in106	385.43	41770.07	**140.96**	**65481.64**
in107	379.15	38781.82	**146.40**	**66245.70**
in108	337.35	43881.51	**161.03**	**74588.51**
in109	336.89	42001.62	**144.71**	**62492.66**
in110	320.84	38632.49	**149.01**	**65171.19**
in111	320.50	43831.12	**157.34**	**72969.16**
in112	317.32	40026.36	**151.40**	**66671.67**
in113	323.59	40135.40	**165.26**	**68901.96**
in114	372.81	40685.71	**160.00**	**64190.63**
in115	417.56	39978.49	**148.03**	**62052.25**
in116	341.92	39152.03	**162.54**	**64849.85**
in117	352.71	41790.51	**152.85**	**66466.39**
in118	301.28	41559.48	**159.06**	**69239.96**
in119	314.40	38458.14	**153.84**	**63968.32**
in120	341.26	40275.37	**166.82**	**68587.41**

Table 3 MA versus GA on some REL-1000-1000 instances

Instances	GA		MA	
	t	*sol*	*t*	*sol*
in201	697.65	56640.60	**98.26**	**77499.82**
in202	693.14	59029.76	**106.68**	**90464.19**
in203	562.29	59476.80	**102.28**	**86239.21**
in204	732.71	57671.10	**97.40**	**81969.046**
in205	573.98	59915.07	**91.26**	**82469.19**
in206	627.01	58674.13	**93.99**	**86881.42**
in207	667.75	60383.29	**100.90**	**91033.51**
in208	646.34	63052.38	**101.29**	**83667.76**
in209	655.09	59333.98	**96.42**	**81966.65**
in210	547.09	64762.35	**97.78**	**85079.98**
in211	866.21	58568.23	**90.78**	**79746.14**
in212	593.75	57672.75	**103.45**	**81061.38**
in213	690.20	59964.10	**101.56**	**83549.21**
in214	719.62	61385.50	**95.06**	**81935.32**
in215	624.50	60232.70	**102.48**	**83663.13**
in216	621.57	59365.35	**100.93**	**83286.63**
in217	792.65	62209.26	**90.34**	**83125.25**
in218	571.62	61195.15	**105.06**	**86936.78**
in219	734.35	62981.18	**93.35**	**88054.21**
in220	544.40	56908.39	**104.35**	**86937.85**

Table 4 MA versus GA on some of the REL-500-1000 instances

Instances	GA		MA	
	t	*sol*	*t*	*sol*
in401	1193.89	56437.68	**37.07**	**72948.07**
in402	1272.06	56637.00	**37.20**	**71454.78**
in403	1299.01	57024.78	**38.81**	**74843.96**
in404	1088.39	61123.14	**38.78**	**78761.68**
in405	1030.96	58852.75	**39.29**	**72674.25**
in406	1318.40	58714.53	**38.09**	**71791.03**
in407	1021.79	58239.19	**40.95**	**73935.28**
in408	1348.82	59185.08	**39.07**	**72580.04**
in409	1342.28	54950.59	**36.28**	**68724.53**
in410	1005.54	59764.76	**41.90**	**71791.57**
in411	1204.06	57763.20	**38.76**	**71200.55**
in412	1078.70	58708.49	**37.17**	**75292.63**
in413	1072.39	56028.32	**40.95**	**73350.87**
in414	1231.54	60838.76	**41.26**	**77146.36**
in415	1163.98	57543.26	**36.32**	**71926.73**
in416	915.70	58341.97	**39.81**	**72520.66**
in417	1050.37	58246.95	**39.29**	**74680.99**
in418	1232.98	57528.04	**40.00**	**71404.84**
in419	1037.37	56596.05	**38.45**	**70472.84**
in420	1240.54	60112.81	**37.65**	**71381.02**
in421	918.68	58766.12	**38.78**	**75694.94**
in422	983.56	61292.74	**37.36**	**72850.90**
in423	1377.53	60477.81	**36.17**	**68134.35**
in424	1030.62	60185.16	**43.26**	**73196.15**
in425	874.73	56802.95	**41.35**	**73258.59**
in426	1224.12	59085.87	**39.67**	**74524.80**
in427	1114.67	62091.74	**36.96**	**73147.95**
in428	986.89	61519.22	**38.64**	**76554.58**
in429	1015.65	58450.56	**39.87**	**75540.96**
in430	974.25	57753.19	**39.60**	**76264.92**

instances. MA finds better quality solutions in shorter time than GA.

Table 5 summarizes the results found by MA and GA on some instances of the group REL-1500-1500. Here too, we can see that MA gives better solutions than GA in shorter CPU time.

Table 6 shows the results of MA and GA on some instances of the group REL-1500-1000. We can remark that the solutions found by MA are better than those found by GA.

From Tables 2, 3, 4, 5 and 6, it can be seen that GA usually fails to find good solutions to the WDP problems for all the checked instances. MA showed a same behavior, it always outperforms GA in both solution quality and efficiency for the different groups of instances.

Figures 5 and 6 confirmed the effectiveness of MA when compared to GA. MA succeeds to find better solutions than

Table 5 MA versus GA on some of the REL-1500-1500 instances

Instances	GA		MA	
	t	*sol*	*t*	*sol*
in601	1489.40	73665.13	**110.62**	**99044.32**
in602	1810.56	76006.38	**114.18**	**98164.23**
in603	1685.07	71585.28	**110.71**	**94126.96**
in604	1627.37	71958.50	**110.60**	**103568.86**
in605	1634.68	71348.06	**122.40**	**102404.76**
in606	1656.29	72505.09	**107.79**	**104346.07**
in607	1625.37	72162.60	**113.26**	**105869.44**
in608	1625.46	76189.79	**109.15**	**95671.77**
in609	1581.18	71664.87	**111.12**	**98566.94**
in610	1572.06	72393.14	**120.17**	**102468.60**
in611	1513.62	73978.10	**107.98**	**98974.64**
in612	1383.70	73874.42	**122.81**	**106056.07**
in613	1520.17	74326.29	**120.14**	**93289.85**
in614	1238.04	68594.13	**122.51**	**97510.72**
in615	1596.39	73182.27	**108.67**	**101770.70**
in616	1449.82	77085.42	**109.65**	**100169.53**
in617	1467.43	78579.03	**114.98**	**100653.88**
in618	1373.21	71538.31	**120.71**	**102378.27**
in619	1491.25	72949.51	**115.00**	**97306.30**
in620	1334.39	75816.58	**115.79**	**102951.68**

Table 6 MA versus GA on some of the REL-1500-1000 instances

Instances	GA		MA	
	t	*sol*	*t*	*sol*
in501	1624.84	64961.36	**107.82**	**79132.03**
in502	1707.18	56954.75	**108.71**	**80340.76**
in503	1450.79	59161.13	**114.15**	**83277.71**
in504	1662.53	59691.51	**116.11**	**81903.02**

those of GA in less CPU time. This difference in performances between MA and GA is due mainly to the inherent premature convergence of the GA algorithm.

The effectiveness of MA is due to the good combination between diversification and intensification which allows a better exploration of the search space and then locate good solutions.

3.3.2 Comparison of MA with Casanova and SAGII

Table 7 shows the results of Casanova, SAGII and MA on some WDP instances where *sol* corresponds to the solution quality found by each algorithm and *t* to the CPU time of the algorithm given in second.

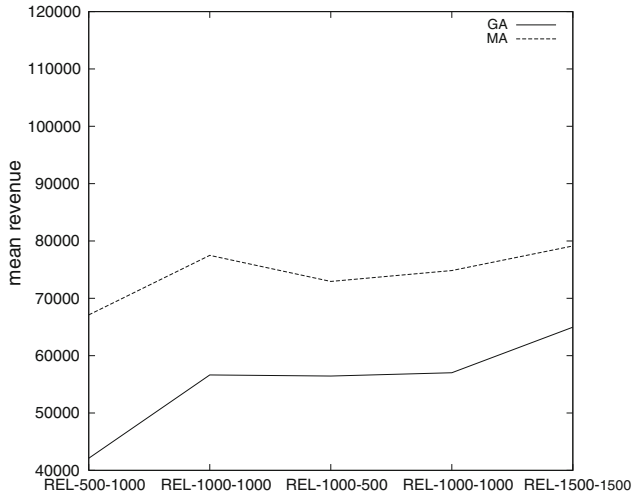

Fig. 5 A comparison of the solution quality between GA and MA

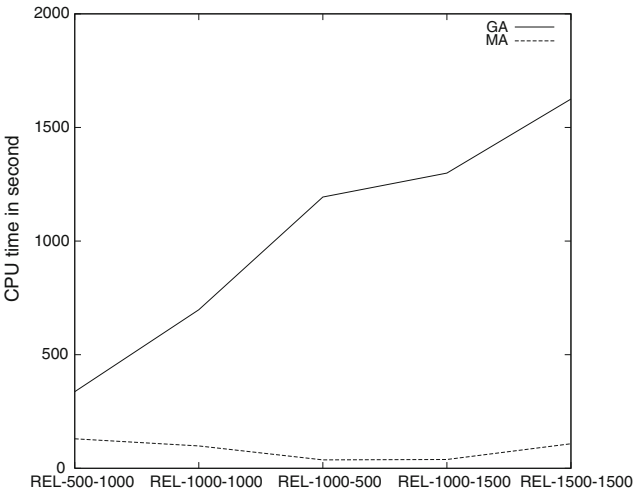

Fig. 6 A comparison of CPU time between GA and MA

The numerical results show that MA performs better than Casanova. It finds better solutions with less CPU time on all the checked instances.

The SAGII method has a good performance and is better than MA on some of the instances, and MA is better than SAGII on some other instances. They seem to be complementary and in average their performances compare on these instances.

3.3.3 A further comparison of MA with Casanova and SAGII

Here we summarizes the results of the methods on the 500 instances of the five groups.

Table 8 (respectively Table 9) shows the numerical results where the column μ_X corresponds to the arithmetic average revenue obtained by the method X on the 100 instances of

Table 7 Casanova, SAGII and MA on some WDP instances

Instances	Casanova		SAGII		MA	
	sol	*t*	*sol*	*t*	*sol*	*t*
in202	52048.73	113.55	86179.64	45.19	**90464.19**	106.68
in208	51340.27	111.16	83500.82	44.80	**83667.76**	101.29
in227	54998.44	117.16	**86747.23**	44.58	83667.76	101.29
in207	51003.39	132.92	88513.37	44.58	**91033.51**	100.90
in501	53992.12	164.94	**85101.43**	71.05	79132.03	107.82
in525	58365.02	164.66	**88086.29**	67.56	85143.13	109.96
in593	58527.76	171.72	**82046.16**	69.03	80304.07	106.89
in594	55821.04	171.80	82341.22	68.11	**86112.90**	103.56
in600	56001.82	174.25	**83772.91**	67.09	82555.91	95.46
in664	65543.41	161.11	**104346.07**	90.73	102905.25	123.81
in688	64962.00	166.88	**106056.08**	90.95	103742.53	114.81
in694	70140.69	170.56	105699.93	91.09	**108114.12**	107.87
in699	65026.40	171.70	103252.95	90.42	**103762.70**	120.71
in700	62404.80	160.98	**105462.71**	91.22	101510.20	117.42

Table 8 MA versus Casanova

Test set	#*ins*	Casanova		MA		$\delta\%$
		Time	μ	Time	μ	
REL-500-1000	100	119.46	37053.78	159.30	**66544.93**	44.32
REL-1000-500	100	57.74	51248.79	38.30	**73562.89**	30.33
REL-1000-1000	100	111.42	51990.91	96.37	**84199.99**	38.25
REL-1000-1500	100	168.24	56406.74	105.66	**80173.42**	29.64
REL-1500-1500	100	165.92	65661.03	113.31	**101035.52**	35.01

Table 9 MA versus SAGII

Test set	#*ins*	SAGII		MA		$\delta\%$
		Time	μ	Time	μ	
REL-500-1000	100	38.06	64922.02	477.22	**67520.23**	3.9
REL-1000-500	100	24.46	73922.10	101.12	**74149.49**	0.3
REL-1000-1000	100	45.37	83728.34	281.63	**84926.39**	0.5
REL-1000-1500	100	68.82	**82651.49**	297.35	80805.98	−0.2
REL-1500-1500	100	91.78	101739.64	321.27	**102234.80**	0.4

each group, the column *time* gives the average time in second and δ is given by the expression $(\mu_{MA} - \mu_{Casanova})/\mu_{MA}$ (respectively $(\mu_{MA} - \mu_{SAGII})/\mu_{MA}$).

We recall that SAGII uses the pre-processor that had been used in Guo et al. (2006) and Sandholm et al. (2001). This pre-processing phase usually reduce the search space and improves the efficiency of the method. We believe that adding a such pre-processing to MA should improve its efficiency.

The results of Table 8 show that MA finds better solutions than Casanova in shorter time and we can see that the difference between the performance of MA and Casanova is even great. MA always gives a 29–44% improvement to Casanova.

To compare MA with SAGII, we have increased the parameter values of MA as follows: we consider a collection C of five high quality individuals and 10 diversified ones, a population of 500 individuals, a number of 500 generations and 500 iterations for the local search process. The results are given in Table 9.

Although the sophisticate Branch and Bound and the pre-processing tools used in SAGII, we can see that MA improves slightly (0.2–4%) in solution quality the SAGII method. But SAGII remains efficient on these instances and is faster than MA.

To further illustrate the results of Tables 8 and 9, we consider the comparative curves of Fig. 7 to show and confirm the

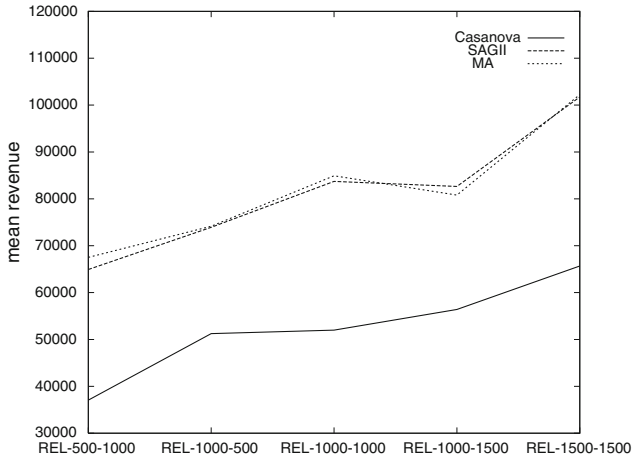

Fig. 7 A comparison of the solution quality of Casanova, SAGII and MA

effectiveness of our approach in reaching good quality solutions for the checked instances. These curves confirm that MA significantly outperforms Casanova on all the checked instances and we observe a slight improvement in favor of MA when comparing it to SAGII.

In overall, the memetic approach outperforms significantly both GA and Casanova on almost all of the checked instances of the WDP problem.

We also remarked that MA compares well with SAGII in average and produces quite similar results in terms of the solution quality. The effectiveness of the memetic approach is due to the good combination of diversification and intensification that leads to a better exploration of the search space.

4 Review of related works

Several methods have been proposed to solve the winner determination problem, among them the primary contribution of Thomas Sandholm (Sandholm 2006). These methods can be divided into two categories: exact and inexact methods.

The exact algorithms, given enough time, permit to find an optimal solution and prove its optimality. The well known exact algorithms for the WDP are based on the Branch-and-Bound method (Sandholm et al. 2001; Sandholm and Suri 2000; Fujishima et al. 1999).

The inexact methods, given enough time, may find optimal solutions, but they cannot be used to prove the optimality of any solution they find. In general, the inexact methods are based on heuristics or metaheuristics and they are shown to be useful in searching solutions for very large instances.

Several exact algorithms to search optimal solutions for the WDP problem have been developed. Among them, the iterative deepening A^* (Sandholm 1999), the branch-on-items (BoI) (Sandholm and Suri 2000), the branch on bids (BoB) (Sandholm and Suri 2000), and the combinatorial auctions BoB (CABoB) (Sandholm et al. 2001). These methods can find reasonable optimal allocation with hundreds of items. The combinatorial auction structural search (CASS) is a branch-and-bound algorithm for the WDP proposed by Fujishima et al. (1999). Leyton-Brown et al. (2000b) proposed combinatorial auctions multi-unit search (CAMUS) which is a new version of the CASS for determining the optimal set of bids in general multi-unit combinatorial auctions. Rothkopf et al. (1998) used a dynamic programming approach to solve the problem, Nisan (2000) proposed a linear programming method and Andersson et al. (2000) proposed another exact algorithm based on integer programming. Holland and O'sullivan used constraint programming to solve a particular Vickrey combinatorial auction (Holland and O'sullivan 2004).

On other hand, different inexact methods are studied for the WDP. Among them Casanova (Hoos and Boutilier 2000) and the hybrid simulated annealing (Guo et al. 2004, 2006). Casanova is a stochastic local search method proposed by Hoos and Boutilier (2000). The principle of the method can be summarized as follows: the algorithm starts with an empty allocation where all items are assigned to a dummy bid and all real bids are unsatisfied. Then, it performs a certain number of local steps that consists in selecting unsatisfied bids to include in the current allocation and in removing any conflicting bids that are originally in the allocation. At each step, the bids are selected as follows: the method selects with a probability wp (walk probability) an unsatisfied bid and with a probability $1 - wp$, it selects a bid 'greedily' by ranking all the bids according to their bid profit divided by the number of items covered by the bid in decreasing order. Therefore, either the highest ranked bid B_1 or the second highest bid B_2 is included into the allocation. Otherwise it inserts B_2 with a probability np (novelty probability) and B_1 with a probability $1 - np$ as it is done in the adaptNovelty for the satisfiability problem (Hoos 2002).

Recently, a new heuristic based on the hybrid simulated annealing, SAGII (Guo et al. 2006) is proposed to find nearly optimal solutions to the WDP problem. The results are very competitive and the new method outperforms dramatically the Casanova method. The proposed simulated annealing method for the WDP makes use of a pre-processing and three local moves. The first move is a Branch-and-Bound applied to a subset of items and bids of the current allocation. The second move is a greedy local search used to find unselected bids which do not conflict with bids that are already in the current allocation and which have a larger greedy value. The greedy value corresponds to the bid profit. To discard conflicting bids, an additional penalty function is introduced. The third move is "1-2 exchange" that randomly picks one or two unselected bids to be added into the current allocation and removes any conflicting bids that are originally

in the current allocation. The method starts with an empty allocation. Then the pre-processing runs once to exclude bids that can lead to suboptimal solutions. The pre-processing may speed up the method. The Branch-and-Bound run with a probability $p_1 = 0.2$. The greedy move is applied with a probability $p_2 = 0.07$, and the "1–2 exchange" move run with a probability $1 - p_1 - p_2$. The process that includes the Branch-and-Bound, the greedy and the "1–2 exchange" moves is repeated until the maximal a number of iterations is reached. For more details about auctions and the WDP, the reader can refer to (McAfee and McMillan 1987; Sandholm 2006; Vries de and Vohra 2003.

5 Conclusion

A memetic algorithm MA for the winner determination problem (WDP) is proposed in this paper and its different components were described. The method MA uses a random key encoding mechanism to generate feasible combinations of bids. The proposed algorithm incorporates a novel selection strategy and a specific crossover operator. The resulting algorithm is enhanced by using a stochastic local search (SLS) component. Our objective is to achieve a good compromise between intensification and diversification in the search process. The combination of the intensification and the diversification strategies leads to a better exploration of the search space and increases the probability to find good solutions. The proposed method is evaluated on several realistic instances and compared with GA, SAGII and Casanova. The obtained results are very encouraging.

To improve our algorithm on quality, new features will be integrated into the proposed algorithm such as the combination of MA and a Branch-and-Bound exact method. Our purpose here, is to find a good compromise when combining exact approaches with inexact ones. To improve on time, pre-processors will be added to exclude bids that can lead to suboptimal solutions.

References

Andersson A, Tenhunen M, Ygge F (2000) Integer programming for combinatorial auction winner determination. In: Proceedings of 4th international conference on multi-agent systems. IEEE Computer Society Press, New York, pp 39–46

Bean JC (1994) Genetics and random keys for sequencing and optimization. ORSA J Comput 6(2):154–160

Boughaci D, Drias H (2005) Taboo Search as an Intelligent Agent for Bid Evaluation. Int J Internet Enter Manage (IJIEM) 3(2):170–186

Boughaci D, Drias H, Benhamou B (2004) Solving Max-SAT problems using a mimetic evolutionary metaheuristic. In: Proceedings of 2004 IEEE CIS 2004, pp 480–484

Burke E, Cowling P, DeCausmaecker Berghe GV (2001) A memetic approach to the nurse rostering problem. Appl Intell 15(3): 199–214

Caponio A, Cascella GL, Neri F, Salvatore N, Sumner M (2007) A fast adaptive memetic algorithm for online and offline control design of pmsm drives. IEEE Trans Syst Man Cybern B 37(1): 28–41

Collins J, Sundareswara R, Gini M, Mobasher B (2000) Bid selection strategies for multi-agent contracting in the presence of scheduling constraints. In: Moukas A, Sierra C, Ygge F (eds) Agent-mediated electronic commerce II. Lecture Notes in AI, vol 1788. Springer, Heidelberg

Dawkins R (1976) The selfish gene. Oxford University Press, Oxford

Franca PM, Mendes A, Moscato P (2001) A memetic algorithm for the total tardiness single machine scheduling problem. Eur J Oper Res 132(1):224–242

Fujishima Y, Leyton-Brown K, Shoham Y (1999) Taming the computational complexity of combinatorial auctions: optimal and approximate approaches. In: Sixteenth international joint conference on artificial intelligence, pp 48–53

Guo Y, Lim A, Rodrigues B, Zhu Y, (2004) Heuristics for a brokering set packing problem. In: Proceedings of eighth international symposium on artificial intelligence and mathematics, pp 10–14

Guo Y, Lim A, Rodrigues B, Zhu Y (2006) Heuristics for a bidding problem. Comp Oper Res 33(8):2179–2188

Hart William E, Krasnogor N, Smith JE (eds) (2005) Recent Advances in Memetic Algorithms. In: Series: studies in fuzziness and soft computing, vol 166

Holland A , O'sullivan B (2004) Towards fast vickrey pricing using constraint programming. Artif Intell Rev 21(3–4):335–352

Hoos HH (2002) An Adaptive Noise Mechanism for WalkSAT. In: Proceedings of the 19 th national conference on artificial intelligence AAAI/IAAI 2002, pp 655–660

Hoos HH, Boutilier C (2000) Solving combinatorial auctions using stochastic local search. In: Proceedings of the 17th national conference on artificial intelligence, pp 22–29

Lau HC, Goh YG (2002) An intelligent brokering system to support multi-agent web-based 4th-party logistics. In: Proceedings of the 14th international conference on tools with artificial intelligence, 2002, pp 54–61

Ishibuchi H, Narukawa K (2004) Some issues on the implementation of local search in evolutionary multi-objective optimization. Proc GECCO 1:1246–1258

Ishibuchi H, Yoshida T, Murata T (2003) 'Balance between genetic search and local search in memetic algorithms for multi-objective permutation flowshop scheduling. IEEE Trans Evolut Comput 7(2):204–223

Leyton-Brown K, Pearson M, Shoham Y (2000a) Towards a universal test suite for combinatorial auction algorithms. In: ACM conference on electronic commerce, pp 66–76

Leyton-Brown K, Tennenholtz M, Shoham Y (2000b) An algorithm for multi-unit combinatorial auctions. In: Proceedings of the 17th national conference on artificial intelligence, Austin, Games-2000, Bilbao, and ISMP-2000, Atlanta

McAfee R, McMillan PJ (1987) Auctions and bidding. J Econ Lit 25:699–738

Moscato P (1989) On evolution, search, optimization, genetic algorithms and martial arts: towards memetic algorithms. In: Caltech concurrent computation program, C3P Report 826

Neri F, Toivanen J, Cascella G, Yew-Soon Ong (2007) An adaptive multimeme algorithm for designing HIV multidrug therapies. IEEE/ACM Trans Comput Biol Bioinf 4(2):264–278

Nisan N (2000) Bidding and allocation in combinatorial auctions. In: Proceedings of ACM conference on electronic commerce (EC'00). ACM SIGecom, ACM Press, Minneapolis, October, pp 1–12

Ong YS, Lim MH, Zhu N, Wong KW (2006) Classification of adaptive memetic algorithms: a comparative study. IEEE Trans Syst Man Cybern B 36(1):141–152

Ong YS, Krasnogor N, Ishibuchi H (eds) (2007) Special Issue on Memetic Algorithms. IEEE Trans Syst Man Cybern B 37(1)

Rothkopf MH, Pekee A, Ronald M (1998) Computationally manageable combinatorial auctions. Manag Sci 44(8):1131–1147

Sandholm T (1999) Algorithms for optimal winner determination in combinatorial auctions. Artif Intell 135(1–2):1–54

Sandholm T (2006) Optimal winner determination algorithms. In: Cramton P et al (ed) Combinatorial auctions. MIT Press, Cambridge

Sandholm T, Suri S, (2000) Improved optimal algorithm for combinatorial auctions and generalizations. In: Proceedings of the 17th national conference on artificial intelligence, pp 90–97

Sandholm T, Suri S, Gilpin A, Levine D (2001) CABoB: a fast optimal algorithm for combinatorial auctions. In: Proceedings of the international joint conferences on artificial intelligence, pp 1102–1108

Tang M, Yao X (2007) A memetic algorithm for VLSI floorplanning. IEEE Trans Syst Man Cybern B 37(1):62–69

Tang J, Lim MH, Ong YS (2007) Diversity-adaptive parallel memetic algorithm for solving large scale combinatorial optimization problems. Soft Comput 11(9):873–888

Vries de S, Vohra R (2003) Combinatorial auctions a survey. INFORMS J Comput 15:284–309

Zhou Z, Ong YS, Lim MH, Lee BS (2007) Memetic algorithm using multi-surrogates for computationally expensive optimization problems. Soft Comput 11(10):957–971