Journal of AI and Data Mining
Vol 5, No 2, 2017, 169-181

Winner Determination in Combinatorial Auctions using Hybrid Ant Colony Optimization and Multi-Neighborhood Local Search

M. B. Dowlatshahi and V. Derhami*

Computer Engineering Department, Yazd University, Yazd, Iran.

Received 02 January 2016; Revised 27 November 2016; Accepted 18 January 2017
**Corresponding author: vderhami@yazduni.ac.ir (V. Derhami).*

Abstract
A combinatorial auction is an auction where the bidders have the choice to bid on bundles of items. The Winner Determination Problem (WDP) in combinatorial auctions is the problem of finding winning bids that maximize the auctioneer's revenue under constraint, where each item can be allocated to at most one bidder. WDP is known as an NP-hard problem with practical applications like electronic commerce, production management, games theory, and resource allocation in multi-agent systems. This has motivated the quest for efficient approximate algorithms in terms of both the solution quality and computational time. This paper proposes a hybrid Ant Colony Optimization with a novel Multi-Neighborhood Local Search (ACO-MNLS) algorithm for solving WDP in combinatorial auctions. Our proposed MNLS algorithm uses the fact that using various neighborhoods in local search can generate different local optima for WDP and that the global optima of WDP is a local optima for a given neighborhood. Therefore, the proposed MNLS algorithm simultaneously explores a set of three different neighborhoods to get different local optima and to escape from the local optima. The comparisons between ACO-MNLS, Genetic Algorithm (GA), Memetic Algorithm (MA), Stochastic Local Search (SLS), and Tabu Search (TS) on various benchmark problems confirm the efficiency of the ACO-MNLS algorithm in terms of both the solution quality and computational time.

Keywords: *Winner Determination Problem, Combinatorial Auctions, Ant Colony Optimization, Multi-Neighborhood Local Search, Combinatorial Optimization.*

1. Introduction

Auctions play a significant role in multi-agent systems, where the auction mechanisms are used for task distribution and resource allocation. The items that are auctioned range from network bandwidth to radio frequencies, and pollution rights. Combinatorial Auction (CA) is a sort of auctions in which bidders (agents) can place bids on combinations of items (goods) rather than only the individual ones. Buyers offer their bids to auctioneer, each bid being defined by a subset of items with a price (bidder's valuation). Two bids are conflicting if they share at least one item. The main advantage of combinatorial auction is that it produces a high economic efficiency [1].

The Winner Determination Problem (WDP) in combinatorial auctions is defined as finding a conflict-free allocation of items that maximize the auctioneer's revenue. WDP is equivalent to the weighted set packing problem, a well-known NP-hard problem [2-4]. From a practical viewpoint, WDP has many applications in electronic commerce, production management, game theory, and resource allocation in multi-agents systems [5-8].

The computational challenge of WDP and its wide practical applications have motivated a variety of algorithms. These algorithms can be classified as either "exact" or "approximate". Exact algorithms can obtain optimal solutions and guarantee their optimality for every instance of WDP. However, it has been shown that for optimization problems that are NP-hard, no polynomial time algorithm exists unless P = NP [3,9]. Therefore, exact algorithms for WDP require exponential time, and

this makes them impractical for most real-world applications. In contrast to exact algorithms, approximate algorithms do not guarantee the optimality of the solutions obtained. In these algorithms, the optimal solution is sacrificed for the sake of obtaining good solutions in a reasonable time [10-12].

Approximate algorithms may be classified into three classes: approximation algorithms, problem-specific heuristics, and metaheuristics. Unlike problem-specific heuristics and metaheuristics, approximation algorithms provide a provable solution quality and run-time bounds. Problem-specific heuristics are problem-dependent and are designed for a particular problem, whereas metaheuristics represent more general approximate algorithms and are applicable to a large variety of optimization problems. Metaheuristics solve complex optimization problems by "exploring" the large solution space and achieve this goal by effectively reducing the size of this space and "exploiting" the reduced space efficiently [10,11,13]. This class of algorithms includes Evolutionary Computation (EC) [14], Ant Colony Optimization (ACO) [15], Greedy Randomized Adaptive Search Procedure (GRASP) [16], Tabu Search (TS) [17], Variable Neighborhood Search (VNS) [18], Iterated Local Search (ILS) [19], Particle Swarm Optimization (PSO) [20], Gravitational Search Algorithm (GSA) [21], etc.

In this paper, we propose a hybrid Ant Colony Optimization with Multi-Neighborhood Search (ACO-MNLS) algorithm for solving WDP. The experimental results obtained by the proposed algorithm are compared with the results of Genetic Algorithm (GA), Memetic Algorithm (MA), Stochastic Local Search (SLS), and Tabu Search (TS). The comparisons confirm the efficiency of ACO-MNLS in terms of solution quality and computational time.

The rest of the paper is organized as what follows. In Section 2, we present the formal definition of WDP and provide an overview of the existing algorithms for WDP. Section 3 provides a review of ACO. In Section 4, the proposed ACO-MNLS algorithm for WDP is presented. Section 5 contains the experimental part of the paper, in which the performance of the proposed approach is evaluated. Finally, in Section 6, conclusion is given.

2. Winner Determination Problem and existing algorithms

2.1. Winner Determination Problem

In this section, we discuss WDP and winner determination algorithms for combinatorial auctions. Let us say that the auctioneer has a set of items, $M = \{1, 2, \ldots, m\}$, to sell, and the buyers propose a set of bids, $B = \{b_1, b_2, \ldots, b_n\}$. A bid is a tuple $b_j = (S_j, p_j)$, where $S_j \subseteq M$ is a set of items and $p_j \geq 0$ is price of b_j, which is a positive real number that shows the value the buyer is willing to pay for bundle S_j. Further, consider a matrix $a_{m\times n}$ having m rows and n columns, where $a_{ij} = 1$ if item i belongs to S_j , $a_{ij} = 0$, otherwise. Finally, the decision variables are defined as follow: $x_j = 1$ if bid b_j is accepted (a winning bid), and $x_j = 0$ otherwise (a losing bid).

WDP is the problem of finding the winning bids that maximize the auctioneer's revenue under the constraint that each item can be allocated to the most bidder. WDP can be modeled as the following integer optimization problem [22]:

$$Maximize \quad \sum_{j=1}^{n} p_j x_j, \tag{1}$$

$$Subject\ to \quad \sum_{j=1}^{n} a_{ij} x_j \leq 1, \quad i \in \{1,\ldots,m\}, \tag{2}$$

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

where, the objective function given in (1) maximizes the auctioneer's revenue that is computed as the sum of prices of the winning bids. The constraints given in (2) mean that the item can be allocated to at most one bidder. The inequality ($a_{ij} x_j \leq 1$) allows that some items could be left uncovered. This is due to the free disposal assumption.

Example 1: Consider a combinatorial auction with a set of five items $M = \{1, 2, 3, 4, 5\}$ to be auctioned and a set of five bids $B = \{b_1, b_2, b_3, b_4, b_5\}$ that are the following:

$b_1 = (\{1, 3\}, 5.5)$
$b_2 = (\{1, 3, 4\}, 15)$
$b_3 = (\{2\}, 1)$
$b_4 = (\{2, 4\}, 12)$
$b_5 = (\{4\}, 8)$
$b_6 = (\{4, 5\}, 10)$.

Note that the combined value of the two bids for the individual items 2 and 4 is lower than the value of the bundle bid for both (b_4), which reflects the complementarity of these items. Let us consider the allocations $A_1 = \{b_2, b_4\}$ and $A_2 = \{b_1,$

b_4}. While A_2 is feasible, A_1 is infeasible because b_2 and b_4 both require item 4. The value for A_2 is 17.5, which is the maximum value over all possible feasible allocations for this problem instance. Under the optimal allocation A_2, bids b_1 and b_4 win, with items 1 and 3 assigned to b_1 and items 2 and 4 assigned to b_4. Note that item 5 remains unassigned under this allocation; there is a feasible allocation that assigns all items to bids ($A_3 = \{b_1, b_3, b_6\}$) but its value is lower than 17.5.

2.2. Existing algorithms

Attempts to exactly solve WDP (under the name of set packing) can be found as early as in the beginning of 1970s [23]. Many studies have appeared ever since. Most exact algorithms are based on the general branch-and-bound (B&B) technique. Some examples include the combinatorial auction structural search (CASS) [2], Combinatorial Auction Multi-Unit Search (CAMUS) [24], BOB algorithm [25], CABOB algorithm [26], and linear programming-based B&B algorithm [27]. Other interesting exact methods for WDP are a branch-and-price algorithm based on a set packing formulation [28], a branch-and-cut algorithm [29], and a dynamic programming algorithm [30]. The general integer programming approach based on CPLEX has been intensively studied in [31,32], showing an excellent performance in many cases. In [33], a clique-based branch-and-bound approach has been introduced for WDP, which relies on a transformation of WDP into the maximum weight clique problem. To ensure the efficiency of the proposed search algorithm, specific bounding and branching strategies using a dedicated vertex coloring procedure and a specific vertex sorting technique has been proposed. In [34], Complete Set Partitioning problem captures the special case of WDP in combinatorial auctions, where bidders place bids on every possible bundle of goods, and the goal is to find an allocation of goods to bidders that maximizes the profit of the auctioneer.

On the other hand, given the intrinsic intractability of WDP, various heuristic algorithms have been devised to handle problems whose optimal solutions cannot be reached by exact approaches. For instance, Casanova [35] is a well-known stochastic local search algorithm that explores the space of feasible allocations (non-overlapping subsets of bids) by adding at each step an unallocated bid and removing from the allocation the bids that are conflicting with the added bid. The selection rule employed by Casanova takes into consideration both the quality and history information of the bid. Casanova has been shown to be able to find high quality solutions much faster than the CASS algorithm [2]. WDP is also modeled as a set packing problem and is solved by a simulated annealing algorithm (SAGII) with three different local move operators: an embedded branch-and-bound move, greedy local search move, and exchange move [32]. SAGII outperforms dramatically Casanova and the CPLEX 8.0 solver for realistic test instances. A memetic algorithm has been proposed by [36], which combines a local search component with a specific crossover operator. The local search component adds at each iteration either a random bid with a probability p or a best bid with the largest profit with probability 1-p, and then removes the conflicting bids from the allocation. This hybrid algorithm reaches excellent results on the tested realistic instances. Other interesting heuristics include greedy algorithm [37], a tabu search algorithm [38], an equilibrium-based local search method [39], and a recombination-based tabu search algorithm [40]. In [41], a new mathematical formulation for WDP (under the name of set packing) and an efficient method for generating near-optimal solution have been proposed. In [42], a mathematical model that aims to maximize the expected economization of procurement has been established and a solution algorithm based on genetic algorithm (GA), where an order encoding scheme is designed and a special repair method is employed to accomplish the translation from the individual encoding to the corresponding solution of WDP, has been proposed. In [43], a stochastic hyper-heuristic (SHH) for combining heuristics for solving WDP has been proposed, in which a new idea is developed for hyper-heuristics by combining choice function and randomness strategies. In [44], an agent learning approach has been proposed for solving WDP, in which a Lagrangian relaxation approach is used to develop an efficient multi-agent learning algorithm. In [45], the authors have presented a metaheuristic approach for the bi-objective WDP, which integrates the greedy randomized adaptive search procedure with a two-stage candidate component selection procedure, large neighborhood search, and self-adaptive parameter setting in order to find a competitive set of non-dominated solutions.

From the above-mentioned review, we observe that the existing (exact and heuristic) methods follow two solution strategies. The first one is to consider directly WDP and design dedicated algorithms. This is the case for most of the

reviewed methods. The second one is to recast WDP as another related problem P and then solved with a solution method designed for P. Examples have been given in [23,32], where WDP is modeled as the set packing problem and in [26, 31], where WDP is reformulated as an integer programming problem and solved by the general CPLEX solver.

2.3. Disadvantages of existing algorithms

The existing exact algorithms to solve WDP [23-34] have an exponential time complexity, and this makes them impractical for most real-world instances of WDP. On the other hand, although heuristic algorithms used to solve WDP [35-45] have a polynomial time complexity, they have a low efficiency and a low effectiveness. To the best of our knowledge, the best results of direct heuristic methods come from a Memetic Algorithm (MA) proposed by [34]. In Section 5, we will see that the proposed ACO-MNLS algorithm outperforms the GA, MA, SLS, and TS algorithms in terms of the computational time, and overcomes the GA, TS, MA, and SLS algorithms in terms of the solution quality in most problems, whereas in the case of other problems, both ACO-MNLS and other algorithms get the same results.

3. Ant Colony Optimization

Ant Colony Optimization (ACO) algorithms are constructive stochastic metaheuristics that make use of a pheromone model and heuristic information on the problem being tackled in order to probabilistically construct solutions. A pheromone model is a set of pheromone trail parameters whose numerical values can be obtained by a reinforcement type of learning mechanism and show the search experience of the algorithm. Therefore, the pheromone model can be used to bias the solution construction over time towards the regions of the solution space containing high quality solutions. Note that the stochastic procedure in ACO permits the ants to explore a much larger number of solutions; meanwhile, the use of heuristic information guides the ants towards the most promising solutions.

Several ACO algorithms for NP-hard problems have been proposed in the literature. Ant System (AS) was proposed as the first ACO algorithm for the well-known Traveling Salesman Problem (TSP) [49]. The Ant Colony System (ACS) [50] and the MAX–MIN Ant System (MMAS) algorithm [51] are among the most successful ACO variants in practice. In order to provide a unifying view to identify the most important aspects of these algorithms, [52], put them in a general framework by defining the ACO metaheuristic. The template of this ACO metaheuristic has been shown in Algorithm (1). After initializing parameters and pheromone trails, the metaheuristic iterates over three phases. At each iteration, a number of solutions are constructed by the ants; these solutions are then improved through a local search (this step is optional), and finally, the pheromone trails are updated.

Algorithm (1): Template of Ant Colony Optimization.

```
Set parameters;
Initialize the pheromone trails;
Repeat
   For each ant Do
        Solution construction using the pheromone trail;
        Solution improvement using local search;
        Update the pheromone trails:
        Evaporation ;
        Reinforcement ;
   Endfor
Until stopping criteria are satisfied.
Output: Best solution found.
```

The *solution construction* is done by a probabilistic rule. Each artificial ant can be considered as a stochastic greedy algorithm that constructs a solution probabilistically by adding solution components to partial ones until a complete solution is derived. This stochastic greedy algorithm takes into account the followings:

Pheromone trails that memorize the patterns of "good" constructed solutions, and will guide the construction of new solutions. The pheromone trails change dynamically during the search to store the obtained knowledge of problem.

Heuristic information that gives more hints about most promising solutions to ants in their decisions to construct solutions.

The *solution improvement* is a local search method that starts with an initial solution and follows moves from the current solution to a neighbor. Many strategies can be used in the selection of a neighbor such as: (1) Best improvement selection strategy, in which the best neighbor (i.e. the neighbor that improves the objective function the most) is selected, (2) First improvement selection strategy, which consists of choosing the first improving neighbor that is better than the current solution, and (3) Random selection strategy, in which a random selection is applied to the neighbors of the current solution. The process of exchanging the current solution with a neighbor is continued until the stopping

criteria are satisfied [53]. Note that solution improvement is an optional component of ACO, although it has been shown that it can improve the performance of ACO when static combinatorial optimization problems are considered. An explanation of the good performance of a combination of ACO with local search can be found in the fact that these two search methods are complementary. An ACO algorithm usually performs a rather coarse-grained search. Therefore, it is a good idea to try and improve its solutions locally.

The *pheromone update* is done using the constructed solutions. A good pheromone updating rule is used in two phases:

An **evaporation phase** that decreases the pheromone trail value. The goal of the evaporation is to escape from premature convergence toward "good" solutions and then to encourage the exploration in the solution space.

A **reinforcement phase** that updates the pheromone trail using constructed solutions. Three different strategies can be used [54]: *off-line pheromone update* [55], *online step-by-step pheromone update* [50], and *online delayed pheromone update* [56]. Among these strategies, the off-line pheromone update is the most popular approach, in which different strategies can be applied: quality-based pheromone update [49], rank-based pheromone update [57], worst pheromone update [58], and elitist pheromone update [51].

4. Proposed algorithm for winner determination

In this section, we present a hybrid ant colony optimization and multi-neighborhood search (ACO-MNLS) algorithm for solving WDP. In addition to common search components in all metaheuristics (e.g. representation of solutions and definition of the objective function), the main components of the proposed ACO-MNLS are pheromone information, solution construction, local search, and pheromone update.

4.1. Solution representation

To design a metaheuristic, representation is necessary to encode each solution of the problem. The representation used in the proposed ACO-MNLS is the binary representation [11]. For a WDP of n bids, a vector $X=\{x_1, x_2, \ldots, x_n\}$ of binary variables x_j may be used to represent a solution:

$$\forall j \in \{1,2,\ldots,n\}, \quad x_j = \begin{cases} 1 & \text{if } b_j \text{ is in solution} \\ 0 & \text{otherwise} \end{cases} . \quad (4)$$

In other words, a solution will be encoded by a vector X of n binary variables, where the jth decision variable of X denotes the presence or absence of the jth bid in the solution. For example, consider a set of five bids $B = \{b_1, b_2, b_3, b_4, b_5\}$ and the feasible allocation $A_1 = \{b_2, b_4\}$ in which bids b_1 and b_4 are won. Figure 1 illustrates a binary representation used by ACO-MNLS for a solution.

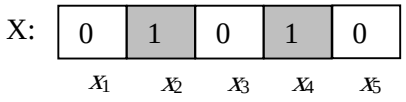

Figure 1. A candidate solution in proposed ACO-MNLS for a WDP with five bids.

4.2. Fitness evaluation

Each metaheuristics must use a fitness evaluation function that associates with each solution of the search space a numeric value that describes its quality. An effective fitness evaluation function must yield better evaluations to solutions that are closer to the optimal solution than those that are farther away. The fitness evaluation function for a given problem is chosen by the problem solver, and it is not given with the problem but is directly related to the specifications for that problem.

Fortunately, the definition of fitness evaluation function for WDP is straightforward. It specifies the originally formulated objective function. The objective function defined in (1) is used to measure the quality of a candidate solution X. Thus for a candidate solution X, its quality is just equal to the sum of the valuations of the winning bids [48]:

$$Fitness(X) = \sum_{j=1}^{n} p_j x_j , \quad (5)$$

where, $X=\{x_1, x_2, \ldots, x_n\}$ is a $1\times n$ matrix, and $P=\{p_1, p_2, \ldots, p_n\}$ is a $1\times n$ matrix in which p_j is the price of b_j.

4.3. Pheromone information

Pheromone information consists of defining a vector of model parameters τ called *pheromone trail parameters*, where pheromone values $\tau_i \in \tau$ should give the relevant information for solution construction. Here, a pheromone τ_j is associated with each bid j (i.e. b_j). Therefore, the pheromone information is represented by a $1\times n$ matrix τ, where each element τ_j of the matrix says the desirability to have the b_j in the solution. The

pheromone matrix is initialized by the same values. During the search, the pheromone is updated to estimate the utility of any bid.

4.4. Solution construction

In addition to the pheromone trails, the main question in the solution construction is concerned with the definition of the problem-specific heuristic to be used in guiding the search. As stated in Section 3, artificial ants can be considered as stochastic greedy algorithms that construct a solution in a probabilistic manner by considering two important parameters: *pheromone trails* and *problem-dependent heuristic information*.

Given an initial arbitrary solution *A*, we define set *C* composed of each bid $b_j = (S_j, p_j)$ such that $S_j \bigcap (\bigcup_{i \in A} S_i) = \phi$. In this case, an ant selects the next bid $b_j \in C$ with the probability:

$$p_j = \frac{(\tau_j)^{\alpha} \times (\eta_j)^{\beta}}{\sum_{k \in C} (\tau_k)^{\alpha} \times (\eta_k)^{\beta}}, \quad (6)$$

where:

- $\eta_j \in \eta$ is the value of problem-specific heuristic for b_j. The problem-specific heuristic information is represented by a $1 \times n$ matrix η , where the value for each element η_j of the matrix is equal to the normalized price of b_j, i.e. $\eta_j = p_j \Big/ \sum_{k=1}^{n} p_k$.
- α and β are the parameters representing the relative influence of the pheromone values and the problem-specific heuristic values. The ACO algorithm will be similar to a stochastic greedy algorithm if we have $\alpha = 0$. In this case, the bids with a large price are more likely to be selected. If $\beta = 0$, only the pheromone trails will guide the search direction. In this case, *stagnation* may occur, in which all ants will construct similar solutions. Hence, a suitable balance must be done in using this kind of information [11].

Note that the process of adding a new bid to the current solution *A* is repeated until set *C* is not empty.

4.5. Local search: Multi-Neighborhood Local Search (MNLS)

Definition of the *neighborhood space* is the common search concept for all local search algorithms. The neighborhood space is defined by an undirected graph $H=(N, E)$ associated with the solution space of the problem, where the nodes in *N* correspond to candidate solutions and the edges in *E* correspond to moves in the neighborhood space, i.e. $(i, j) \in E$ if and only if $i \in N,\ j \in N,\ j \in N(i)$, and $i \in N(j)$, where $N(y)$ denotes the neighbors of a solution $y \in N$. The structure of the neighborhood depends on the target optimization problem. A neighbor solution *y'* for a given solution *y* is constructed by applying a move *m* to the solution *y* using a move operator $\oplus$, denoted by $y' = y \oplus m$. The neighborhood space is called single-neighborhood if for constructing it we use only a one-move operator, and is called multi-neighborhood if for constructing it we use several-move operators [11].

A local search may be seen as a walk in the neighborhood space. A walk is performed by move operators that move from the current solution to another one in the neighborhood space. Here, we define three basic move operators for WDP, denoted by *ADD*, *EXCHANGE*, and *REMOVE*. Suppose an initial arbitrary allocation *A* composed of some non-conflicting bids.

The *ADD*(b_j) move operator consists of adding to *A* a bid $b_j = (S_j, p_j)$ from the set of bids that are excluded from the *A* and have no conflict with bids in *A*. In example 1, let us consider the feasible allocation $A = \{b_1, b_3\}$. There are only two bids b_5 and b_6 that are excluded from the *A* and have no conflict with bids in the *A*. Note that after the *ADD*(b_j) move, the change in the fitness of solution is $+p_j$. Since the value for p_j is always positive, the move gain is always positive for an *ADD* move, and therefore, such a move always leads to an improved neighboring solution.

EXCHANGE(b_i, b_j) move operator consists of exchanging a bid $b_i = (S_i, p_i)$ (from the set of bids that are excluded from the *A* and have no conflict with bids in set $A\text{-}b_j$) with only bid b_j of *A* that have conflict with b_i. In example 1, let us consider the feasible allocation $A = \{b_1, b_3\}$. Bid b_2 is a candidate bid to exchange with bid b_1, and bid b_4 is a candidate bid to exchange with bid b_3. The move gain of the *EXCHANGE*(b_i, b_j) move operator is $p_i - p_j$. Note that the move gain can be either positive or negative for an *EXCHANGE* move. Hence, we can see that an *EXCHANGE* move can increase or decrease the fitness of *A*.

The *REMOVE*(b_j) move operator removes a bid $b_j = (S_j, p_j)$ from the *A*. The move gain of the removed bid b_j is $-p_j$. Note that the move gain is always negative for a *REMOVE* move because p_j is always positive. Hence, we can see that a

REMOVE move always leads to a decrease in the fitness of *A*.

For the three move operators *ADD*, *ECXHANGE*, and *REMOVE*, there is no absolute dominance of one operator over the other ones. Therefore, the best move operator to be applied depends on the current situation. These facts lead us to generate a combined neighborhood space *H*, which corresponds to the union of the three neighborhoods H_1, H_2, and H_3, denoted by $H = H_1 \cup H_2 \cup H_3$. Using this multi-neighborhood, our local search algorithm, i.e. Multi-Neighborhood Local Search (MNLS), at each iteration selects the move with the largest gain among all the *ADD*, *ECXHANGE*, and *REMOVE* moves if the move gain is positive, and selects a random move among all possible moves if the move gain is negative. Note that the MNLS algorithm simultaneously explores a set of three neighborhoods H_1, H_2, and H_3 to get different local optima and to escape from local optima. MNLS uses the fact that using various neighborhoods in local search can generate different local optima and that the global optima is a local optima for a given neighborhood. The template of the MNLS algorithm is shown in Algorithm (2).

Algorithm (2): Template of Multi-Neighborhood Local Search algorithm for WDP.

Input: *X as the initial solution, and maxiter as the maximum iteration of MNLS algorithm.*
For *i = 1 to maxiter* ***Do***
    Generate candidate neighbors to X by three move operators ADD, ECXHANGE, and REMOVE;
    X' = the best neighbor of X;
    If *Fitness(X')-Fitness(X) > 0* ***Then***
        X = X';
    Else
        X = a random neighbor of X;
    Endif
Endfor
Output: *Best solution found.*

4.6. Pheromone update

As stated in Section 3, a general pheromone updating strategy is used in two phases: **evaporation phase** and **reinforcement phase**. Here, we use the classical evaporation method for the pheromone trails so that each pheromone value is reduced by a fixed proportion. For each b_j, its pheromone τ_j will evaporate as follows:

$$\tau_j = (1-\rho)\tau_j, \quad \forall j \in \{1,...,n\}, \tag{7}$$

where, $\rho \in [0, 1]$ shows the reduction rate of the pheromone.

Now the pheromone update method has to be specified. Here, we use elitist pheromone update [35], in which the best solution found so far will increment the pheromone matrix to reinforce exploitation ability of the search. This operation is done by (8):

$$\tau_j = \tau_j + \Delta, \quad \text{if } b_j \text{ is a winner bid}, \tag{8}$$

where, $\Delta = Fitness(X) \Big/ \sum_{k=1}^{n} p_k$, and $P=\{p_1, p_2, \ldots, p_n\}$ is a $1\times n$ matrix, in which p_k is the price of b_k.

4.7. General framework of ACO-MNLS

The pseudo-code of ACO-MNLS is described in Algorithm (3). At first, the initial values for the parameters are determined. After initialization, the main search loop is entered. It is repeated until a maximum number of iterations is satisfied. In the main loop itself, four important phases exist: Solution construction, Solution improvement, pheromone evaporation, and pheromone reinforcement.

Algorithm (3): Template of ACO-MNLS.

Set the value of below parameters:
    the number of ants;
    the initial value of pheromone matrix ;
    the relative influence of the pheromone values, i.e. α;
    the problem-dependent parameter β;
    the reduction rate of the pheromone, i.e. ρ ;
    the maximum number of iterations;
    the maximum iteration of MNLS algorithm;
Repeat
  For *each ant* ***Do***
      Solution construction using the pheromone trail;
      Solution improvement using MNLS algorithm;
      Pheromone evaporation using Eq. (7);
      Pheromone reinforcement using Eq. (8);
  Endfor
Until *maximum number of iterations are satisfied.*
Output: *Best solution found.*

5. Experimental results

In this Section, the performance of the proposed algorithm is measured on several benchmark instances. In order to show the effectiveness of our approach, we compared the ACO-MNLS algorithm with four different approaches for solving the WDP reported in [48], i.e. Stochastic Local Search (SLS), Tabu Search (TS), Genetic Algorithm (GA), and Memetic Algorithm (MA). The structure of this section is as what follows. First we describe the characteristics of the selected benchmarks. Then we present the results obtained from ACO-MNLS for benchmark instances. Finally, we present a comparison of ACO-MNLS with the other four metaheuristics.

5.1. Benchmarks

To evaluate the performance of algorithms on the WDP problem, [59] has created the program

Combinatorial Auction Test Suite (CATS) to generate benchmarks. Recently, [37] has provided new benchmarks of various sizes consisting of up to 1500 items and 1500 bids. The CATS instances are easily solved by CPLEX and CABoB [60]. In this paper, we use the realistic benchmarks by [37] for which CPLEX cannot find the optimal solution in a reasonable period of time. These benchmarks include 500 instances, and are available at the Zhuyi's home page (http://logistics.ust.hk/~zhuyi/instance.zip). These benchmarks are divided into five groups of problems, where each group contains 100 instances given in table 1. In this table, *m* is the number of items and *n* is the number of bids.

5.2. Results and comparisons

In this section, the performance of the proposed ACO-MNLS is measured by applying the proposed algorithm to solve different benchmarks. The proposed ACO-MNLS was implemented in C language and run on a PC with an Intel 2.2 GHz CPU. The ACO-MNLS parameters are fixed on the following values: the number of ants is set to 100, the initial value of pheromone matrix is set to 10, the relative influence of the pheromone values, i.e. α parameter, is set to 0.5, the relative influence of the problem-dependent heuristic values, i.e. β parameter, is set to 5, the reduction rate of the pheromone, i.e. ρ parameter, is set to 0.1, the stopping criterion of ACO-MNLS is satisfied after 200 iterations, and the maximum iteration of MNLS algorithm is set to 50. All of these values for the parameters are obtained experimentally.

Tables 2–6 present the computational results of the ACO-MNLS algorithm in comparison with different metaheuristics reported in [48]. Each table is designed to one of the 5 groups of the REL benchmarks.

In these tables, the first column shows the name of the instance, columns with *sol* caption correspond to the maximum revenue obtained by each algorithm, and columns with *time* caption correspond to CPU time in seconds for each algorithm.

From tables 2-6, it can be observed that the ACO-MNLS algorithm outperforms the GA, MA, SLS, and TS algorithms in terms of computational time in all the REL instances. Also ACO-MNLS outperforms GA in terms of the solution quality in all REL instances and overcomes the TS, MA, and SLS algorithms in most instances, whereas in the case of other instances, both the ACO-MNLS and other algorithms get the same results. The proposed ACO-MNLS is ranked in the first place among five metaheuristics in terms of both the solution quality and computational time. In order to determine the statistical significance of the advantage of ACO-MNLS, *t*-test (all compared with ACO-MNLS) is applied. In the first row of each table, the symbols + and ≈ represent that other methods are statistically inferior to or equal to the proposed algorithm, respectively. The last three rows of each table summarize how many cases ACO-MNLS perform better, similar or worse than the other algorithms. From these results, we can conclude that the ACO-MNLS algorithm dominates the GA [48], MA [48], SLS [48], and TS [48] algorithms in terms of both the solution quality and computational time.

Note that from an optimization viewpoint, ACO-MNLS combine global and local search using ACO to perform exploration, while the MNLS algorithm performs exploitation. ACO ensures that ACO-MNLS can explore new bids that may have not been seen in the search process yet. In fact, ACO makes the entire search space reachable, despite the finite population size. Furthermore, the MNLS algorithm was able to enhance the convergence rate of ACO-MNLS by finely tuning the search on the immediate area of the landscape being considered.

Table 1. Main characteristics of benchmarks used.

Benchmarks	*m*	*n*	Description
REL-1000-500	500	1000	100 instances from in 101 to in 200
REL-1000-1000	1000	1000	100 instances from in 201 to in 300
REL-500- 1000	1000	500	100 instances from in 401 to in 500
REL-1500-1000	1000	1500	100 instances from in 501 to in 600
REL-1500-1500	1500	1500	100 instances from in 601 to in 700

Table 2. Experimental results of proposed ACO-MNLS, GA, MA, SLS, and TS on some instances of REL-1000-500.

Instances	ACO-MNLS		GA		MA		SLS		TS	
	sol	time	sol	time	sol	time	sol	time	sol	time
in101	**69840.07**	**16.84**	42100.71^{+}	336.90	67101.93$^{\approx}$	129.62	66170.61^{+}	23.51	66170.61^{+}	57.86
in102	**70897.46**	**16.02**	39641.22^{+}	432.76	67797.61^{+}	132.18	65466.95^{+}	23.89	64716.31^{+}	63.43
in103	**69791.25**	**15.36**	43376.54^{+}	338.89	66350.99^{+}	133.34	66350.99^{+}	24.79	66350.99^{+}	128.68
in104	**67268.71**	**15.64**	42790.65^{+}	376.37	64618.41^{+}	135.14	**67268.71**$^{\approx}$	22.92	62524.23^{+}	120.56
in105	**69834.28**	**17.14**	40841.21^{+}	331.31	66376.83^{+}	153.96	67268.71$^{\approx}$	22.92	62524.23^{+}	120.56
in106	**66436.08**	**13.48**	41770.07^{+}	385.43	65481.64$^{\approx}$	140.96	63479.26^{+}	22.37	64591.70$^{\approx}$	129.42
in107	**69182.25**	**14.28**	38781.82^{+}	379.15	66245.70^{+}	146.40	66245.70^{+}	23.18	63972.62^{+}	128.51
in108	**74588.51**	**16.14**	43881.51^{+}	337.35	**74588.51**$^{\approx}$	161.03	71505.66^{+}	24.01	68776.34^{+}	119.84
in109	**66239.28**	**13.56**	42001.62^{+}	336.89	62492.66^{+}	144.71	61751.22^{+}	22.20	64343.07$^{\approx}$	80.98
in110	**67395.07**	**14.28**	38632.49^{+}	320.84	65171.19$^{\approx}$	149.01	64083.64^{+}	23.25	60275.66^{+}	115.31
Average	69147.30	15.28	41381.78	357.59	66622.55	142.64	65959.15	23.30	64424.58	106.52
Rank	1	1	5	5	2	4	3	2	4	3
Better	-	-	10	-	6	-	8	-	8	-
Similar	-	-	0	-	4	-	2	-	2	-
Worse	-	-	0	-	0	-	0	-	0	-

Table 3. Experimental results of proposed ACO-MNLS, GA, MA, SLS, and TS on some instances of REL-1000-1000.

Instances	ACO-MNLS		GA		MA		SLS		TS	
	sol	time	Sol	time	Sol	time	sol	time	sol	time
in201	**81557.74**	**6.10**	56640.60^{+}	697.65	77499.82^{+}	98.26	56640.60^{+}	697.65	77499.82^{+}	98.26
in202	**90464.19**	**7.32**	59029.76^{+}	693.14	**90464.19**$^{\approx}$	106.68	59029.76^{+}	693.14	**90464.19**$^{\approx}$	106.68
in203	**86239.21**	**7.00**	59476.80^{+}	562.29	**86239.21**$^{\approx}$	102.28	59476.80^{+}	562.29	**86239.21**$^{\approx}$	102.28
in204	**87075.42**	**6.98**	57671.10^{+}	732.71	81969.05^{+}	97.40	57671.10^{+}	732.71	81969.05^{+}	97.40
in205	**82469.19**	**6.16**	59915.07^{+}	573.98	**82469.19**$^{\approx}$	91.26	59915.07^{+}	573.98	**82469.19**$^{\approx}$	91.26
in206	**86881.42**	**6.32**	58674.13^{+}	627.01	**86881.42**$^{\approx}$	93.99	58674.13^{+}	627.01	**86881.42**$^{\approx}$	93.99
in207	**91033.51**	**6.38**	60383.29^{+}	667.75	**91033.51**$^{\approx}$	100.90	60383.29^{+}	667.75	**91033.51**$^{\approx}$	100.90
in208	**91782.20**	**7.22**	63052.38^{+}	646.34	83667.76^{+}	101.29	63052.38^{+}	646.34	83667.76^{+}	101.29
in209	**81966.65**	**6.82**	59333.98^{+}	655.09	**81966.65**$^{\approx}$	96.42	59333.98^{+}	655.09	**81966.65**$^{\approx}$	96.42
in210	**87569.19**	**6.52**	64762.35^{+}	547.09	85079.98$^{\approx}$	97.78	64762.35^{+}	547.09	85079.98$^{\approx}$	97.78
Average	86703.87	6.68	59893.95	640.31	84727.08	98.63	59893.95	640.31	84727.08	98.63
Rank	1	1	4	4	2	2	4	4	2	2
Better	-	-	10	-	3	-	10	-	3	-
Similar	-	-	0	-	7	-	0	-	7	-
Worse	-	-	0	-	0	-	0	-	0	-

Table 4. Experimental results of proposed ACO-MNLS, GA, MA, SLS, and TS on some instances of REL-500-1000.

Instances	ACO-MNLS		GA		MA		SLS		TS	
	sol	time	sol	time	sol	time	sol	time	sol	time
in401	**77417.48**	**3.52**	56437.68^{+}	1193.89	72948.07^{+}	37.07	72948.07^{+}	5.67	68485.81^{+}	44.14
in402	**74469.07**	**3.94**	56637.00^{+}	1272.06	71454.78^{+}	37.20	71454.78^{+}	5.79	72820.03$^{\approx}$	23.57
in403	**74843.96**	**3.80**	57024.78^{+}	1299.01	**74843.96**$^{\approx}$	38.81	**74843.96**$^{\approx}$	6.01	**74843.96**$^{\approx}$	34.15
in404	**78761.68**	**3.84**	61123.14^{+}	1088.39	**78761.68**$^{\approx}$	38.78	**78761.68**$^{\approx}$	6.12	73385.62^{+}	16.85
in405	**74899.12**	**4.02**	58852.75^{+}	1030.96	72674.25$^{\approx}$	39.29	72674.25$^{\approx}$	6.04	72674.25$^{\approx}$	15.90
in406	**71791.03**	**3.56**	58714.53^{+}	1318.40	**71791.03**$^{\approx}$	38.09	**71791.03**$^{\approx}$	5.87	**71791.03**$^{\approx}$	37.12
in407	**73935.28**	**4.16**	58239.19^{+}	1021.79	**73935.28**$^{\approx}$	40.95	73278.66$^{\approx}$	6.35	71578.48^{+}	15.57
in408	**77018.73**	**3.98**	59185.08^{+}	1348.82	72580.04^{+}	39.07	72580.04^{+}	5.95	70144.19^{+}	27.37
in409	**73188.62**	**3.36**	54950.59^{+}	1342.28	68724.53^{+}	36.28	67177.35^{+}	5.48	67177.35^{+}	25.48
in410	**73791.66**	**4.24**	59764.76^{+}	1005.54	71791.57^{+}	41.90	71791.57^{+}	6.37	72791.68$^{\approx}$	14.01
Average	75011.66	3.84	58092.95	1192.11	72950.52	38.74	72730.14	5.97	71569.24	25.42
Rank	1	1	5	5	2	4	3	2	4	3
Better	-	-	10	-	5	-	5	-	5	-
Similar	-	-	0	-	5	-	5	-	5	-
Worse	-	-	0	-	0	-	0	-	0	-

Table 5. Experimental results of proposed ACO-MNLS, GA, MA, SLS, and TS on some instances of REL-1500-1000.

Instances	ACO-MNLS		GA		MA		SLS		TS	
	sol	Time	sol	time	sol	time	sol	time	sol	time
in501	**84165.23**	**6.28**	64961.36^{+}	1624.84	79132.03^{+}	107.82	77140.72^{+}	15.62	82216.35$^{\approx}$	98.71
in502	**83163.66**	**6.16**	56954.75^{+}	1707.18	80340.76^{+}	108.71	78574.26^{+}	15.98	74127.61^{+}	120.82
in503	**83277.71**	**5.98**	59161.13^{+}	1450.79	**83277.71**$^{\approx}$	114.15	79554.65^{+}	15.99	77005.81^{+}	114.11
in504	**83947.13**	**5.66**	59691.51^{+}	1662.53	81903.02$^{\approx}$	116.11	81903.02$^{\approx}$	16.48	81903.02$^{\approx}$	155.54
Average	83638.43	6.02	60192.19	1611.34	81163.38	111.70	79293.16	16.02	78813.20	122.30
Rank	1	1	5	5	2	3	3	2	4	4
Better	-	-	4	-	2	-	3	-	2	-
Similar	-	-	0	-	2	-	1	-	2	-
Worse	-	-	0	-	0	-	0	-	0	-

Table 6. Experimental results of proposed ACO-MNLS, GA, MA, SLS, and TS on some instances of REL-1500-1500.

Instances	ACO-MNLS		GA		MA		SLS		TS	
	sol	time	sol	time	sol	time	sol	time	sol	time
in601	**105286.68**	**5.88**	73665.13^{+}	1489.40	99044.32^{+}	110.62	96255.53^{+}	15.54	97473.85^{+}	100.76
in602	**101150.89**	**5.22**	76006.38^{+}	1810.56	98164.23^{+}	114.18	95328.21^{+}	15.71	93873.31^{+}	155.34
in603	**96628.98**	**5.22**	71585.28^{+}	1685.07	94126.96$^{\approx}$	110.71	94126.96$^{\approx}$	15.48	92568.61^{+}	137.95
in604	**106127.19**	**5.50**	71958.50^{+}	1627.37	103568.86^{+}	110.60	103568.86^{+}	15.59	92869.78^{+}	96.70
in605	**106273.50**	**6.02**	71348.06^{+}	1634.68	102404.76^{+}	122.40	98799.71^{+}	17.36	95787.59^{+}	175.14
in606	**105218.21**	**5.42**	72505.09^{+}	1656.29	104346.07$^{\approx}$	107.79	104346.07$^{\approx}$	15.60	104346.07$^{\approx}$	334.12
in607	**105869.44**	**5.52**	72162.60^{+}	1625.37	**105869.44**$^{\approx}$	113.26	100417.40$^{\approx}$	15.89	98674.39^{+}	267.79
in608	**99541.75**	**5.38**	76189.79^{+}	1625.46	95671.77^{+}	109.15	95671.77^{+}	15.26	91554.61^{+}	95.62
in609	**104602.39**	**5.26**	71664.87^{+}	1581.18	98566.94^{+}	111.12	98566.94^{+}	16.76	96652.44^{+}	103.10
in610	**109008.35**	**6.12**	72393.14^{+}	1572.06	102468.60^{+}	120.17	99975.09^{+}	17.57	99975.09^{+}	146.03
Average	103970.70	5.54	72947.88	1630.74	100423.20	113.00	98705.65	16.08	96377.57	161.26
Rank	1	1	5	5	2	3	3	2	4	4
Better	-	-	10	-	7	-	7	-	9	-
Similar	-	-	0	-	3	-	3	-	1	-
Worse	-	-	0	-	0	-	0	-	0	-

6. Conclusions

A hybrid Ant Colony Optimization with a novel Multi-Neighborhood Local Search (ACO-MNLS) algorithm was proposed for solving Winner Determination Problem (WDP) in combinatorial auctions. Our proposed MNLS algorithm used the fact that using various neighborhoods in local search could generate different local optima for WDP and that the global optima of WDP was a local optima for a given neighborhood. Therefore, in the proposed MNLS algorithm, a set of three different neighborhoods was simultaneously explored to get different local optima and to escape from local optima. To the best of our knowledge and the research in the literature, no study has been done to solve WDP with combining general-purpose Ant Colony Optimization (ACO) metaheuristic and problem-specific Multi-Neighborhood Local Search (MNLS) algorithm.

The performance of the proposed algorithm was evaluated in terms of solution quality and computational time by several well-known benchmarks. Its performance was compared with four different metaheuristics for solving WDP, i.e. Stochastic Local Search (SLS), Tabu Search (TS), Genetic Algorithm (GA), and Memetic Algorithm (MA). The experimental results confirmed that the proposed ACO-MNLS outperformed the current best performing WDP metaheuristics in terms of both the solution quality and computational efficiency.

A first step toward extending this paper would be to hybrid the proposed MNLS algorithm in other swarm and evolutionary algorithms. Secondly, the MNLS algorithm could be changed to simultaneously explore a set of other different neighborhoods. Finally, the proposed approach could be adopted for solving Multi-objective WDP (MOWDP) [45].

References

[1] Parsons, S., Rodriguez-Aguilar, J. A. & Klein, M. (2011). Auctions and bidding: A guide for computer scientists. ACM Computing Surveys, vol. 43, no. 2, pp. 1-10.

[2] Fujishima, Y., Leyton-Brown, K. & Shoham, Y. (1999). Taming the computational complexity of combinatorial auctions: optimal and approximate approaches. Sixteenth international joint conference on artificial intelligence, Stockholm, Sweden, 1999.

[3] Garey, M. & Johnson, D. (1979). Computers and Intractability: A Guide to the Theory on NP-Completeness. W.H. Freeman and Co. Publishers, New York.

[4] Sandholm, T. (2006). Optimal Winner Determination Algorithms, In: Cramton, P. (Eds.), Combinatorial Auctions. MIT Press, pp. 337-368.

[5] Abrache, J., Crainic, T. G., Gendreau, M. & Rekik, M. (2007). Combinatorial auctions. Annals of Operations Research, vol. 153, no. 1, pp. 131–164.

[6] Fontanini, W. & Ferreira, P. A. V. (2014). A game-theoretic approach for the web services scheduling problem. Expert Systems with Applications, vol. 41, no. 10, pp. 4743–4751.

[7] Ray, A. K., Jenamani, M. & Mohapatra, P. K. J. (2011). Supplier behavior modeling and winner determination using parallel MDP. Expert Systems with Applications, vol. 38, no. 5, pp. 4689–4697.

[8] Vries, S. & Vohra, R. (2003). Combinatorial auctions: a survey. INFORMS Journal on Computing, vol. 15, pp. 284-309.

[9] Lipton, R. J. (2010). The P=NP Question and Godel's Lost Letter, Springer.

[10] Safaee, B. & Kamaleddin Mousavi Mashhadi, S. K. (2017). Optimization of fuzzy membership functions via PSO and GA with application to quad rotor. Journal of AI and Data Mining, vol. 5, no. 1, pp. 1-10.

[11] Talbi, E. G. (2009). Metaheuristics: From Design to Implementation. John Wiley & Sons.

[12] Dowlatshahi, M. B., Nezamabadi-pour, H. & Mashinchi, M. (2014). A discrete gravitational search algorithm for solving combinatorial optimization problems. Information Sciences, vol. 258, pp. 94–107.

[13] AllamehAmiri, M., Derhami, V. & Ghasemzadeh, M. (2013). QoS-Based web service composition based on genetic algorithm. Journal of AI and Data Mining, vol. 1, no. 2, pp. 63-73.

[14] Holland, J. H. (1975). Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor, MI.

[15] Dorigo, M. (1992). Optimization, learning and natural algorithms. Dissertation, Politecnico di Milano, Italy.

[16] Kirkpatrick, S., Gelatt, C. D. & Vecchi, M. P. (1983). Optimization by simulated annealing. Science, vol. 220, pp. 671–680.

[17] Feo, T. A. & Resende, M. G. C. (1989). A probabilistic heuristic for a computationally difficult set covering problem. Operations Research Letters, vol. 8, pp. 67–71.

[18] Glover, F. & Laguna, M. (1997). Tabu Search. Kluwer Academic Publishers.

[19] Mladenovic, M. & Hansen, P. (1997). Variable neighborhood search. Computers and Operations Research, vol. 24, pp. 1097–1100.

[20] Stützle, T. (1999). Local search algorithms for combinatorial problems: Analysis, algorithms and new applications. Dissertation, Germany.

[21] Kennedy, J. & Eberhart, R. (1999). Particle swarm optimization. In: Proceedings of the IEEE International Conference on Neural Networks, Piscataway, N.J., pp. 1942–1948.

[22] Rashedi, E., Nezamabadi-pour, H. & Saryazdi, S. (2009). GSA: A Gravitational Search Algorithm. Information Sciences, vol. 179, pp. 2232–2248.

[23] Sandholm, T. (1999). Algorithms for optimal winner determination in combinatorial auctions. Artificial Intelligence, vol. 135, pp. 1–54.

[24] Padberg, M. W. (1973). On the facial structure of set packing polyhedra. Mathematical Programming, vol. 5, no. 1, pp. 199–215.

[25] Leyton-Brown, K., Shoham, Y. & Tennenholz, M. (2000). An algorithm for multi-unit combinatorial auctions. In: Proceedings of the 7th international conference on artificial intelligence pp. 56–61.

[26] Sandholm, T. & Suri, S. (2003). BOB: Improved winner determination in combinatorial auctions and generalizations. Artificial Intelligence, vol. 145, pp. 33–58.

[27] Sandholm, T., Suri, S., Gilpin, A. & Levine, D. (2005). CABOB: A fast optimal algorithm for winner determination in combinatorial auctions. Management Science, vol. 51, no. 3, pp. 374–390.

[28] Nisan, N. (2000). Bidding and allocation in combinatorial auctions. In: Proceedings of the 2nd ACM conference on electronic commerce, pp. 1–12.

[29] Gunluk, O., Laszlo, L. & de Vries, S. (2005). A branch-and-price algorithm and new test problems for spectrum auctions. Management Science, vol. 51, no. 3, pp. 391–406.

[30] Escudero, L. F., Landete, M. & Marin, A. (2009). A branch-and-cut algorithm for the winner determination problem. Decision Support Systems, vol. 46, no. 3, pp. 649–659.

[31] Rothkopf, M. H., Pekec, A. & Harstad, R. M. (1998). Computationally manageable combinatorial auctions. Management Science, vol. 44, no. 8, pp. 1131–1147.

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

[33] Guo, Y., Lim, A., Rodrigues, B. & Zhu, Y. (2006). Heuristics for a bidding problem. Computers & Operations Research, vol. 33, no. 8, pp. 2179-2188.

[34] Wu, Q. & Hao, J.K. (2016). A clique-based exact method for optimal winner determination in combinatorial auctions. Information Sciences, vol. 334, pp. 103-121.

[35] Michalak, T., Rahwan, T., Elkind, E., Wooldridge, M. & Jennings, N.R. (2016). A hybrid exact algorithm for complete set partitioning. Artificial Intelligence, vol. 230, pp. 14-50.

[36] Hoos, H. H. & Boutilier, C. (2000). Solving combinatorial auctions using stochastic local search. In: Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence, pp. 22-29.

[37] Boughaci, D., Benhamou, B. & Drias, H. (2009). A memetic algorithm for the optimal winner determination problem. Soft Computing - A Fusion of Foundations, Methodologies and Applications, vol. 13, pp. 905-917.

[38] Lau, H. C. & Goh, Y. G. (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, pp. 54–61.

[39] Boughaci, D., Benhamou, B. & Drias, H. (2010). Local search methods for the optimal winner determination problem in combinatorial auctions. Journal of Mathematical Modelling and Algorithms, vol. 9, no. 2, pp. 165–180.

[40] Tsung, C., Ho, H. & Lee, S. (2011). An equilibrium-based approach for determining winners in combinatorial auctions. In: Proceedings of the 9th IEEE international symposium on parallel and distributed processing with applications, pp.47–51.

[41] Sghir, I., Hao, J. K., Ben Jaafar, I. & Ghedira, K. (2014). A recombination-based tabu search algorithm for the winner determination problem. In: Legrand, P., et al. (Eds.), AE 2013. Lecture notes in computer science, vol. 8752, pp. 157–169.

[42] Nguyen, T. D. (2014). A fast approximation algorithm for solving the complete set packing problem. European Journal of Operational Research, vol. 237, no. 1, pp. 62-70.

[43] Wang, N. & Wang, D. (2014). Model and algorithm of winner determination problem in multi-item E-procurement with variable quantities. In: The 26th Chinese Control and Decision Conference, pp. 5364-5367.

[44] Boughaci, D. & Lassouaoui, M. (2014). Stochastic Hyper-Heuristic for the Winner Determination Problem in combinatorial auctions. In: Proceedings of the 6th International Conference on Management of Emergent Digital EcoSystems, pp. 62-66.

[45] Hsieh, F. S. & Liao, C. S. (2014). Multi-agent Learning for Winner Determination in Combinatorial Auctions. In: International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, pp. 1-10.

[46] Buer, T. & Kopfer, H. (2014). A Pareto-metaheuristic for a bi-objective winner determination problem in a combinatorial reverse auction. Computers & Operations Research, vol. 41, pp. 208-220.

[47] Holte, R. (2001). Combinatorial Auctions, Knapsack Problems, and Hill-Climbing Search. In: Stroulia, E. & Matwin, S. (Eds.), Advances in Artificial Intelligence, vol. 2056 of Lecture Notes in Computer Science, pp. 57-66.

[48] Schwind, M., Stockheim, T. & Rothlauf, F. (2003). Optimization heuristics for the combinatorial auction problem. In: Evolutionary Computation, 2003. CEC '03. The 2003 Congress on, vol. 3, pp. 1588-1595.

[49] Boughaci, D. (2013). Metaheuristic approaches for the winner determination problem in combinatorial auction. In: Yang, X.S. (Eds.), Artificial Intelligence, Evolutionary Computing and Metaheuristics, vol. 427 of Studies in Computational Intelligence, pp. 775-791.

[50] Dorigo, M., Maniezzo, V. & Colorni, A. (1996). Ant System: Optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics – Part B, vol. 26, no. 1, pp. 29–41.

[51] Dorigo, M. & Gambardella, L. M. (1997). Ant Colony System: A cooperative learning approach to the traveling salesman problem. IEEE Transactions on Evolutionary Computation, vol. 1, no. 1, pp. 53–66.

[52] Stützle, T. & Hoos, H. H. (2000). MAX–MIN Ant System. Future Generation Computer Systems, vol. 16, no. 8, pp. 889–914.

[53] Dorigo, M., Caro, G. D. & Gambardella, L. M. (1999). Ant algorithms for discrete optimization. Artificial Life, vol. 5, no. 2, pp. 137–172.

[54] Aarts, E. H. L. & Lenstra, J. K. (1997). Local Search in Combinatorial Optimization. J. Wiley & Sons, Chichester, UK.

[55] Corne, D., Dorigo, M. & Glover, F. (1999). New Ideas in Optimization. McGraw-Hill.

[56] Merkle, D. & Middendorf, M. (2005). Swarm intelligence. In Search Methodologies. Springer, pp. 401–435.

[57] Maniezzo, V. (1999). Exact and approximate nondeterministic tree-search procedures for the quadratic assignment problem. INFORMS Journal on Computing, vol. 11, no. 4, pp. 358–369.

[58] Bullnheimer, B., Hartl, R. F. & Strauss, C. A. (1999). New rank based version of the ant system: A computational study. Central European Journal for Operations Research and Economics, vol. 7, no. 1, pp. 25–38.

[59] Cordon, O., Fernandez, I., Herrera, F. & Moreno, L. (2000). A new ACO model integrating evolutionary algorithms concepts: The best-worst ant system. In: 2nd International Workshop on Ant Algorithms, Brussels, Belgium, pp. 22–29.

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

[61] 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.

نشریه هوش مصنوعی و داده کاوی

تعیین برنده در حراج‌های ترکیبیاتی با استفاده از ترکیب الگوریتم‌های بهینه‌ساز جمعیت مورچگان و جستجوی محلی چندهمسایگی

محمدباقر دولتشاهی و ولی درهمی*

گروه مهندسی کامپیوتر، دانشکده فنی، دانشگاه یزد، یزد، ایران.

ارسال ۲۰۱۶/۰۱/۰۲؛ بازنگری ۲۰۱۶/۱۱/۲۷؛ پذیرش ۲۰۱۷/۰۱/۱۸

چکیده:

یک حراج ترکیبیاتی، حراجی است که در آن پیشنهادکنندگان باید پیشنهادات خود را برای خریدن یک بسته از عناصر ارائه دهند. مسئله‌ی تعیین برنده در حراج ترکیبیاتی عبارت است از مسئله‌ی پیدا کردن پیشنهاداتی که سود حراج کننده‌ها را تحت این محدویت که هر عنصر فقط می‌تواند به یک پیشنهاد دهنده تخصیص یابد، بیشینه می‌کند. مسئله‌ی تعیین برنده یک مسئله‌ی ان پی-سخت است که کاربردهای عملی زیادی مانند تجارت الکترونیک، مدیریت تولید، نظریه بازی‌ها، و تخصیص منابع در سیستم‌های چند عامله دارد. این موضوع انگیزه‌ی اصلی ما برای جستجوی الگوریتم‌های تقریبی کارامد بر حسب کیفیت راه‌حل و زمان محاسباتی است. این مقاله یک الگوریتم ترکیبی با استفاده از ترکیب کردن الگوریتم بهینه‌ساز جمعیت مورچگان با یک الگوریتم جستجوی محلی چندهمسایگی جدید برای حل مسئله‌ی تعیین برنده در حراج ترکیبیاتی ارائه می‌دهد. الگوریتم جستجوی محلی چندهمسایگی پیشنهادی از این واقعیت بهره می‌برد که استفاده از چندین همسایگی در الگوریتم جستجوی محلی باعث پیدا کردن بهینه‌های محلی مختلفی می شود، و بهینه‌ی سراسری برای مسئله‌ی تعیین برنده خود یک بهینه‌ی محلی برای یک همسایگی خاص است. بنابراین الگوریتم جستجوی محلی چندهمسایگی پیشنهادی به طور همزمان از سه همسایه‌ی مختلف برای فرار از بهینه‌ی محلی استفاده می‌کند. مقایسه‌ی میان الگوریتم پیشنهادی با الگوریتم ژنتیک، الگوریتم ممتیک، جستجوی محلی تصادفی، و جستجوی تابو بر روی مسائل محک مختلف کارامدی الگوریتم پیشنهادی را بر حسب کیفیت راه حل و زمان محاسباتی نشان می‌دهد.

کلمات کلیدی: مسئله تعیین برنده، حراج‌های ترکیبیاتی، بهینه‌ساز جمعیت مورچگان، جستجوی محلی چندهمسایگی، بهینه‌سازی ترکیبیاتی.