Heuristic and Algorithm for Mixed Multi-Unit Combinatorial Auctions

JianCong Bai, Zhong Ming
School of Software, Shenzhen University, Shenzhen, Guangdong, 518060, PRC
szubic@szu.edu.cn

Abstract

Mixed multi-unit combinatorial auctions provide two types of bids as XOR-bids and OR-bids, which make them popular for resource allocation in marketplace. However, the problem of maximizing the revenue of auctioneers is a NP-complete problem. In this paper, we propose an improved partheno-genetic algorithm for solving this problem. The algorithm repeals crossover operators and implements the functions of crossover and mutation by partheno-genetic operators. New heuristics and a worst-removed operator are designed for improving the profit of the solution. Simulation results show that the algorithm has great searching speed and achieves good performance in large scale problems.

Keywords: multi-unit combinatorial auctions, winner determination problem, partheno-genetic algorithm, OR-bids, XOR-bids

1. Introduction

Multi-unit combinatorial auctions are a popular type of auctions, which allow bidders bidding multiple units on one item in a bid and bidding on combinations of items. With the wildly applications of e-commerce, multi-unit combinatorial auctions (MUCA) provide fully automated electronic negotiation for a variety of task and resource allocation in complex marketplace, for example, transformability relationships among goods, online double auction and so on [1-4].

Mixed multi-unit combinatorial auctions (MMUCA) provide two types of bids as XOR-bids and OR-bids [5-7]. These two new kinds of bids allow bidders to submit additive or exclusive bids over collection of combinations. For example, a bidder wants to bid (A, B) or (B, C) on three items (A, B, C) in an auction, but he doesn't want them all. In basic MUCA, he can choose only one item combination and submits (A, B) or (B, C) in a single bid. In MMUCA, he can use XOR-bids to express his preference on item combinations by submitting a XOR-bids as “(A, B) XOR (B, C)”. XOR-bids and OR-bids help bidders expressing their preference more exactly than before, and achieve more efficient allocations than traditional MUCA.

In combinatorial auctions (CA), finding the revenue maximizing set of winning bids is the first difficult challenge [8], called winner determination problem (WDP). The WDP in CA is a complex computational problem and NP-complete [9]. The researches for WDP in single-unit combinatorial auctions (SUCA) are plenty. Sandholm and we have solved the problem in SUCA well [6, 10-12]. However, the researches for WDP in MUCA are few. Some researches have tried to solve the problem in different ways. Vinyals provided an algorithm to generate artificial data and an integer pro-
programming implementation for MMUCA in CPLEX [7]. Gonen and Lehmann investigated the use of branch-and-bound techniques and characterized the theoretically optimal method for sorting bids [13]. Leyton-Brown developed a branch-and-bound algorithm for the problem, which incorporated a specialized dynamic programming procedure [14].

In this paper, we propose an improved partheno-genetic algorithm (PGA) to find solutions with a worst-removed operator and three partheno-genetic operators instead of complex operators of crossover and mutation. Then new heuristics are designed for evaluating the bids and applied in the worst-removed operator to improve the profit of the solution. At last, we test the PGA in two data distribution and the PGA has great searching speed and achieves good performance in large scale problems.

2. Problem Description

In MMUCA, each bidder $i$ submits a bid set as $B_i^{type}$, not a single bid. In $B_i^{type}$, type denotes that it is a XOR-bids or OR-bids. $B_i^{type}$ contains a group of bids, for example, $B_i^{XOR} = \{b_1, b_2\}$ indicates that bidder $i$ wants one of $\{b_1, b_2\}$ to be chosen, but not both. And then $B_i^{OR} = \{b_1, b_2\}$ indicates that bidder $i$ wants $b_1, b_2$, or both to be chosen.

Bid $b_{ij}$ denotes that bidder $i$ bids on an item combination. And $b_{ij}$ is a pair $(s_{ij}, p_{ij})$, where $p_{ij}$ is the price of the bid, and $s_{ij}$ is a vector containing the numbers of requested units of each item. $s_{ij} = (r_{i1}, r_{i2}, \ldots, r_{ik}, \ldots)$, $r_{ik}$ is the number of requested units of item $k$ in bid $b_{ij}$, where $0 \leq r_{ik} \leq q_k$. If bid $b_{ij}$ doesn’t bid on item $k$, then $r_{ik} = 0$.

Assume that there is one seller (or several sellers acting in concert) and multiple bidders. The goal is to maximize the seller’s revenue. The model of MMUCA is as follows:

**MMUCA:**

\[
\max \sum_{j=1}^{n} \sum_{j=1}^{m} p_{ij} x_{ij} 
\]

s.t.

\[ \sum_{i=1}^{n} \sum_{j=1}^{m} x_{ij} r_{ijk} \leq q_k \quad k = 1, 2, \ldots, m \]  
\[ \sum_{j=1}^{m} x_{ij} \leq 1 \quad \text{type} = \text{XOR}, \quad i = 1, \ldots, n \]  
\[ x_{ij} \in \{0, 1\} \]  

In above model, $m$ is the number of items to be auctioned, $n$ is the number of bidders in MMUCA. $q_k$ is the available units of item $k$. In equation (3), $\left| B_i^{type} \right|$ is the number of bids in the bid set.

Equation (2), (3) and (4) are constraints of the model, equation (4) denotes that if bid $b_{ij}$ is chosen into the solution, $x_{ij} = 1$ else $x_{ij} = 0$. Equation (2) ensures that the sum of requested unit of each item should be not greater than the number of available. Equation (3) ensures that no more than one bid in XOR-Bids can be chosen.

3. Our Algorithm

The problem of MMUCA belongs to combinatorial optimization problems. Genetic algorithms (GA) are capable of solving these problems, such as traveling salesman problem, flow-shop problem and so on [15, 16]. Different from GA, partheno-genetic algorithms (PGA) use partheno-genetic operators to implement the genetic operation of crossover and mutation. Partheno-genetic operators are SWAP, REVERSE and INSERT [17, 18]. These operators replace the special crossover operators for solving combinatorial optimization problems, such as PMX, OX.
and CX [19]. And they can restrain immature convergence phenomenon and don’t require initial population to be varied [17]. To increase the profit of the solution, we design a worst-removed operator to remove the worst gene and try to generate a better solution. And we design some heuristics to evaluate the bids and apply them into the worst-removed operator for finding the worst bid in the current solution. We call the PGA with heuristics and the worst-removed operator as improved partheno-genetic algorithm (IPGA).

3.1. Genetic operation

In our IPGA, the natural number string is used as the gene representation. A chromosome represents an allocation. The fitness is the sum price of the bids in the solution.

Each chromosome is generated from a particular sequence of all bids. We generate a random sequence \( Q \) on all bids firstly, and then take the bid from \( Q \) one by one, until the end of \( Q \). If the bid is not violated the equation (2) and (3) in the model MMUCA, add it into the solution. The procedure of chromosome generation has been studied in [11].

In genetic operation, each new population is generated by three partheno-genetic operators working on chromosomes in the current population. These three partheno-genetic operators have been studied in [10, 11]

3.2. Heuristics for Bids Evaluation

A feasible solution is generated by selecting bids one by one from \( Q \) in turn until the end of \( Q \). It is obviously that the bids in the front of \( Q \) have more chance to be chosen than the bids in the back. To approach the optimal solution, good bids should be placed in the front of \( Q \) as far as possible while the bad bids should be closer to the end of \( Q \). Profit is a direct way to evaluate a bid. Different from SUCA [10], the evaluation of bids in MUCA need to consider the price of the bid, the number of item bid on, the requested unit of each item, the available unit of each item, and relations between all these factors. The evaluating strategy is that the fewer item a bid bidding on, the smaller proportion of the request unit and available unit of each item, and the higher the price is, the better the bid is. The evaluation function is as follows:

**Heuristic \( C(b_{ij})_k \)**, the proportion of the request unit in bid \( b_{ij} \) to the available unit of each item

\[
C(b_{ij})_k = \frac{q_k}{r_{ijk}} \quad k = 1,2,...,m, \quad r_{ijk} \neq 0
\]  

**Heuristic \( G(b_{ij})_k \)**, \( p_{ij} \) is the product of the price of bid \( b_{ij} \) and the heuristic denotes the contribution of \( b_{ij} \) on each item

\[
G(b_{ij})_k = p_{ij} \times C(b_{ij})_k \quad k = 1,2,...,m, \quad r_{ijk} \neq 0
\]

**Heuristic \( W(b_{ij}) \)**, \( N(b_{ij}) \) is the number of the items, which bid \( b_{ij} \) bid on, and the heuristic denotes the average contribution of \( b_{ij} \) on all items, which it bid on.

\[
W(b_{ij}) = \frac{\sum_{k=1}^{m} G(b_{ij})_k}{N(b_{ij})} \quad k = 1,2,...,m
\]

3.3. Worst-removed operator

To each chromosome, the worst-removed operator is used to find the worst gene, and remove it. Equally, it is also used to find out the worst bid in a solution and remove the worst bid out. The usage of the operator is not only finds the worst bid, but also increases the profit of the solution. Let \( X \) be the current solution, and it is generated from the bid sequence \( Q \). The procedure of the worst-removed operator working on \( X \) is as follows:

**Step1.** Calculate \( W(b_{ij}) \) for all bids in \( X \)
Step 2. Select the bid with minimum \( W(b_{ij}) \) as the worst bid \( b_{wst} \).

Step 3. Move \( b_{wst} \) to the end of \( Q \), and then generate new bid sequence \( Q' \).

Step 4. Generate a new solution \( X' \) from new sequence \( Q' \).

Step 5. If the profit of \( X' \) is higher than \( X \), replace \( X \) with \( X' \) and output \( X' \) as a new chromosome, else keep \( X \) no change.

3.4. Algorithm Steps

After all works have been introduced as above, the procedure of the IPGA for MMUCA is as follows:

Step 1. Initialize parameters as Table 1.

Step 2. Produce initial random population.

Step 3. If the stopping criterion is satisfied, go to Step 7; else continue.

Step 4. Perform three partheno-genetic operations on the \( k \)th generation \( A_k \) and generates \( B_k \).

Step 5. Perform worst-removed operation on \( B_k \) and generate \( C_k \).

Step 6. Perform the roulette wheel proportional selection and quintessence selecting strategy on \( C_k \), and obtain the next generation \( A_{k+1} \), then go to Step 3.

Step 7. Stop and output the best solution.

Table 1 Global Parameters

| NG | Number of generations |
| PN | Population size |
| \( P_s \) | Probability of SWAP operator |
| \( P_r \) | Probability of REVERSE operator |
| \( P_i \) | Probability of INSERT operator |
| \( P_w \) | Probability of Worst-Removed |

4. Experimental Results

Our IPGA is programmed in C language and run on two different data distributions: Random and Weighted. In Random distribution, the items, the requested unit and the price of each bid are set randomly. In Weighted distribution, the price of a bid is associated with the number of items and requested units. The more items contained in the bid and the more units requested by the bid, the higher its price is. In all data, XOR-bids must contain at least two bids.

We test IPGA on execution time and deviation rate. Deviation rate denotes the deviation between the average result \( V_{aver} \) and the max result \( V_{max} \). Each instance runs ten times, and deviation rate is calculated as following formula:

\[
\text{Deviation rate} = \left( \frac{V_{max} - V_{aver}}{V_{max}} \right) \times 100\%
\]

Each bar in the following histograms represents an average over 10 problem instances. Each instance is test with different random seeds.

4.1. Execution Time

Fig. 1 shows that IPGA has good searching speed. In large problem size as 50 items and 500 bids, the execution time of IPGA is less than 1 minute. The time cost on weighted distribution is higher than that on random distribution in large problem size.

4.2. Deviation Rate

Fig. 2 shows that IPGA achieves small deviation rate on random distribution. Even in large problem size, the deviation rate of IPGA is less than 2%. Fig. 3 shows that when the problem size becomes large, deviation rate on weighted distribution is higher than 2%. In large problem size,
deviation rate on weighted distribution is less than 4%.

![Graph](image1)

Fig. 2: Deviation rate of IPGA on random distribution

![Graph](image2)

Fig. 3: Deviation rate of IPGA on weighted distribution

5. Conclusion and Further Work

In this paper, we propose IPGA for solving the problem in MMUCA. The IPGA uses partheno-genetic operation and restrain the immature convergence phenomenon. Heuristics are designed for bids evaluation and applied in the worst-removed to increase the profit of the solution. It is well known that the problem in CA is related with knapsack problems (KP) closely [20]. In the further, we will apply our algorithm for solving other problems related with KP.

Acknowledgements

The authors would like to thank the referees for useful comments, which help to improve the paper. This paper is partially supported by National Science Foundation of China (60673122), National Science Foundation of GuangDong Province of China (7301315), and Technology Plan Project of Shenzhen City under Grant No. 200731.


