Winner Determination in Combinatorial Reverse Auctions

Shubhashis Kumar Shil, Malek Mouhoub, and Samira Sadaoui

Abstract. Since commercially efficient, combinatorial auctions are getting more interest than traditional auctions. However, winner determination problem is still one of the main challenges of combinatorial auctions. In this paper, we propose a new method based on genetic algorithms to address two important issues in the context of combinatorial reverse auctions: determining the winner(s) in a reasonable processing time and reducing the procurement cost. Indeed, not much work has been done using genetic algorithms to determine the winner(s) specifically for combinatorial reverse auctions. To evaluate the performance of our method, we conducted several experiments comparing our proposed method with another method related to determining winner(s) in combinatorial reverse auctions. The experiment results clearly demonstrate the superiority of our method in terms of processing time and procurement cost.

1 Introduction

An auction is a market scenario in which bidders compete for item(s). In traditional auctions, an individual item is auctioned separately, which leads to an inefficient allocation and processing time [7, 10]. Combinatorial auctions have been proposed to improve the efficiency of bid allocation by allowing bidders to bid on multiple items [7, 10]. These auctions provide a combinatorial allocation that minimizes the procurement cost and running time [5, 7, 10]. They have been used in various real-world situations [1] such as resource allocation with real-time constraints [10], sensor management [9, 11], supply chain management [12] and computer grids [2]. A combinatorial auction problem is actually a winner determination problem [4]. Winner determination is still one of the main challenges of combinatorial auctions [10]. Indeed, determining the winner(s) in combinatorial

Shubhashis Kumar Shil · Malek Mouhoub · Samira Sadaoui
Department of Computer Science, University of Regina, Regina, SK, Canada
e-mail: {shil200s,mouhoubm,sadaouis}@uregina.ca

M. Ali et al. (Eds.): *Contemporary Challenges & Solutions in Applied AI*, SCI 489, pp. 35–40.
DOI: 10.1007/978-3-319-00651-2_5

auctions is a complex problem and it has been shown to be NP-complete [5, 7]. However, applying combinatorial auctions to procurement scenario [13] is cost–saving [7]. Many algorithms have been developed to solve combinatorial auction problems, e.g. Hsieh and Tsai presented a Langrangian heuristic method [7], and Sitarz introduced Ant algorithms and simulated annealing [7]. Furthermore, many research works have been carried out to figure out the efficient way to solve winner determination in combinatorial auctions. Most of the proposed algorithms restrict the bundles on which bids can be submitted in order to solve the problem optimally but these restrictions introduce economic inefficiencies [7]. Some algorithms find optimal solutions but are very slow; others avoid restrictions but allow bidding on a small number of items [7]. We are interested in combinatorial reverse auctions in which we consider the procurement of a single unit of multi-items. In our auction, there is one buyer and several sellers who compete according to the buyer's requirements. First the buyer announces his demand (multiple items) in the auction system. Then the interested sellers register for that auction and bid on a combination of items. Genetic Algorithms (GAs) are successful to solve many combinatorial optimization problems [7]. GAs are powerful search techniques consisting of selection, crossover and mutation methods [3]. Sometimes simple crossover and mutation operators produce inappropriate chromosomes. To avoid this problem, special or modified crossover and mutation operators are defined [1]. GAs can terminate anytime as required and the current best chromosome can be the best solution. Nevertheless, not much work has been done by using GAs to solve winner determination problem in the context of combinatorial reverse auctions. To the best of our knowledge, only one research paper [7] employed GAs to tackle this problem. However, the method proposed in [7] needs comparatively many generations and a considerable amount of time to produce good solutions. In this paper, our research goal is twofold: (1) solve the winner determination problem in combinatorial reverse auctions in a reasonable processing time, and (2) reduce the procurement cost with fewer generations. For this purpose, we define a new GA-based method that uses two repairing techniques to repair infeasible chromosomes as well as a modified two-point crossover operator that is capable of distributing the solutions and preventing a premature convergence. Furthermore, we conduct several experiments by comparing our proposed method with the one defined in [7]. The experimental results clearly demonstrate the superiority of our method in terms of processing time and procurement cost.

2 Proposed GA-Based Method

In Fig. 1, we define our GA-based method that we name GACRA (Genetic Algorithms for Combinatorial Reverse Auctions). Assume there are m items and n sellers. So the number of bid items combination is 2^m-1 and we use m×n bits to represent each chromosome. In case of m=2 and n=3, a chromosome represented by 100100 means that seller 1 bids for only item A (first two bits 10), seller 2 bids

for only item B (next two bits 01) and seller 3 bids for no items (last two bits 00). To generate bid prices, we consider random values between 200 and 500 for each item for each seller. In Step 2, the initial chromosomes are generated randomly. To avoid redundancy, the RemoveRedundancy function ensures exactly one selection of a particular item from all sellers in each chromosome. To avoid emptiness, the RemoveEmptiness function guarantees at least one selection of every item in each chromosome. So, we repair infeasible chromosomes using these two functions. Our RemoveRedundancy function works with the following steps.

1. For each chromosome, selects bits for each seller.
2. Tests bits for one seller to verify if item(s) are selected and stores this information.
3. Continue testing bits of next sellers; if item(s) are already selected by the previous seller then converts the current bit value to 0.

Our RemoveEmptiness function works with the following steps.

1. For each chromosome, selects bits for each seller.
2. Tests bits for all sellers and stores the information of the non- selected item(s).
3. Continue converting bit value to 1 until all item(s) are selected.

Input: number of bid items: m, number of sellers: n, number of generations: δ, crossover rate: α, mutation rate: β
Terminology: number of chromosomes in a population: n× m, number of combinations of bid items: 2^m-1
Features: RemoveRedundancy method, RemoveEmptiness method, modified two-point crossover
Consideration: minimize bid price in optimal running time
Output: winner(s)

1. generate bid prices for each combination of bid items for each seller
2. generate initial chromosomes
3. check feasibility and remove redundancy and emptiness, if necessary

Loop until number of generations achieved

{

4. compute fitness values of chromosomes
5. select chromosomes with gambling-wheel disk selection method
6. compute fitness values of chromosomes
7. generate child chromosomes from parents with modified two-point crossover considering crossover rate, α
8. mutate chromosomes considering mutation rate, ß
9. check feasibility and remove redundancy and emptiness, if necessary
10. compute fitness values of chromosomes
11. determine better chromosomes from both initial and new chromosomes of each generation

}

12. return winner(s) with minimum bid price in optimal running time

Fig. 1 Pseudo code of GACRA

In step 4, fitness value of chromosome is generated. We propose the following fitness function to calculate the fitness value of every chromosome. Since the motivation of this research work is to minimize the procurement cost for the buyer, our fitness function for Chromosome X_i is defined as follows.

$$F(X_i) = \frac{1}{\sum_{s=1}^{n}\sum_{C=1}^{2^m-1} b_s(C) \times x_s(C)} \qquad / x_s(C) \in \{0,1\} \tag{1}$$

where $b_s(C)$ represents a bid for the item combination C submitted from the s^{th} seller; $x_s(C)$ is 1 when the item combination C is selected for the s^{th} seller and 0 otherwise.

We use gambling-wheel disk selection method [4] to select chromosomes for cross-over operation. In step 7, the crossover operation is performed. A child chromosome takes two portions from one parent and one portion from another parent in two-point crossover. In our modified two-point crossover operation, the basic idea is same but the direction of taking portions from parents is different. The first child takes portions in forward direction but the second child takes it in reverse order. It creates positive effect to increase diversity in the solution spaces to allow all bidders to get more chances to be selected. In some cases, after removing redundancy and emptiness, some bidders get deprived but this modification of crossover gives them a chance again. This will prevent the procedure to converge prematurely. Then, the procedure will move to mutation operation as indicated in our algorithm. In step 9, RemoveRedundancy and RemoveEmptiness functions remove redundancy and emptiness respectively. In step 11, the procedure selects the better chromosomes among the initial and new chromosomes of the generation based on fitness values. Since genetic algorithm is called anytime algorithm, our procedure can be stopped anytime and it produces the best solution. The entire process is repeated until the termination condition is fulfilled, which is here the number of generations. In step 12, the procedure returns the winner(s). The solution is not improving in all generations but in our procedure we always preserve the current winner. So there is no chance to produce worse solution than the previous generation.

3 Experiment

We have conducted several experiments to determine the winner in combinatorial reverse auctions by using our method GACRA. We also compare GACRA with the technique presented in [7] that we call CRA (Combinatorial Reverse Auctions). We have implemented GACRA as well as CRA as described in [7] in Java. These two methods are both executed on an AMD Athlon (tm) 64 X2 Dual Core Processor 4400+ with 3.43 GB of RAM and 2.30 GHz of processor speed. We have used the following parameters and settings for the experiments: Chromosome Encoding: Binary String; Selection: Gambling-Wheel Disk; Crossover: Modified

Two-point; Crossover Rate: 0.6; Mutation Rate: 0.01 and Termination Condition: Generation Number.

In the first experiment, we measure the required time of our proposed method and compare it with CRA. In Fig. 2, we show the required time (in milliseconds) versus the number of generations for both GACRA and CRA. This is the average required time of 20 runs. From the comparison we can see that our proposed method needs less processing time. This happens because of two reasons: (1) we represent the chromosome with less number of bits, and (2) we keep the calculation of fitness value simple.

We have also done some comparative experiments on the procurement cost and report the results in Fig. 2. Since our procedure always maintains feasible solutions and never accepts redundant bid item, it is able to produce good solutions from the very first generations. Moreover, it keeps producing better solutions in consecutive generations.

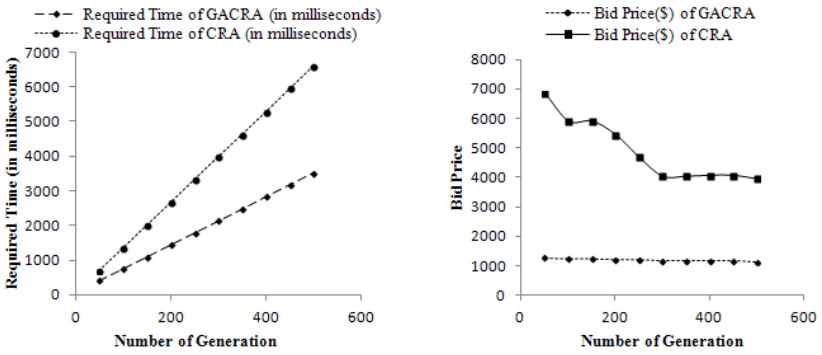

Fig. 2 Left figure - required time (in milliseconds) vs number of generation; right figure – bid price vs number of generation

In addition, we assess the processing time for GACRA by varying the number of sellers and keeping the number of items fixed to 2, 4, 6 and 8. Finally we assess the processing time for GACRA by varying the number of items and keeping the number of sellers fixed to 20, 40, 60 and 80. As the number of bits required to represent the chromosomes of our algorithm directly depends on both the number of items and the number of sellers, so with increasing any of these, the required running time also increases. Due to using less number of bits to represent chromosome, required running time of GACRA is less than that of CRA. In CRA fitness function tries to mitigate the effect of allowing redundancy but in GACRA fitness function is simple and has no additional task. In CRA repair function only removes emptiness but in GACRA RemoveRedundancy, RemoveEmptiness functions and Modified two-point crossover operator remove redundancy, emptiness and early convergence respectively. Due to not allowing redundancy, GACRA is able to minimize procurement costs from very early generations.

4 Conclusion and Future Work

Motivated to save significant time and reduce the procurement cost, this paper solves the problem of winner determination in combinatorial reverse auctions. With the help of the two repairing method, the modified two-point crossover and a careful selection of operator of GAs, it is notable that our method can produce optimal solutions in a reduced processing time. As it is obvious that parallel GAs are capable of providing the solution more efficiently [6, 8], the future target of this research is to determine the winner by using the most efficient methods of parallel GAs. Another future direction is to determine the winner(s) according to multiple attributes of items.

References

1. Easwaran, A.M., Pitt, J.: An Agent Service Brokering Algorithm for Winner Determination in Combinatorial Auctions. In: ECAI, pp. 286–290 (2000)
2. Das, A., Grosu, D.: A Combinatorial Auction-Based Protocols for Resource Allocation in Grids. In: 19th IEEE International Parallel and Distributed Processing Symposium (2005)
3. Goldberg, D.E., Deb, K.: A Comparative Analysis of Selection Schemes Used in Genetic Algorithms, pp. 69–93 (1991); edited by G.J.E. Rawlins
4. Gong, J., Qi, J., Xiong, G., Chen, H., Huang, W.: A GA Based Combinatorial Auction Algorithm for Multi-Robot Cooperative Hunting. In: International Conference on Computational Intelligence and Security, pp. 137–141 (2007)
5. Zhang, L.: The Winner Determination Approach of Combinatorial Auctions based on Double Layer Orthogonal Multi-Agent Genetic Algorithm. In: 2nd IEEE Conference on Industrial Electronics and Applications, pp. 2382–2386 (2007)
6. Nowostawski, M., Poli, R.: Parallel Genetic Algorithm Taxonomy. In: Third International Conference on Knowledge-Based Intelligent Information Engineering Systems, pp. 88–92 (1999)
7. Patodi, P., Ray, A.K., Jenamani, M.: GA Based Winner Determination in Combinatorial Reverse Auction. In: Second International Conference on Emerging Applications of Information Technology, pp. 361–364 (2011)
8. Abbasian, R., Mouhoub, M.: An Efficient Hierarchical Parallel Genetic Algorithm for Graph Coloring Problem. In: 13th Annual Genetic and Evolutionary Computation Conference, pp. 521–528. ACM, Dublin (2011)
9. Mullen, T., Avasarala, V., Hall, D.L.: Customer-Driven Sensor Management. IEEE Intelligent Systems 21(2), 41–49 (2006)
10. Avasarala, V., Polavarapu, H., Mullen, T.: An Approximate Algorithm for Resource Allocation using Combinatorial Auctions. In: International Conference on Intelligent Agent Technology, pp. 571–578 (2006)
11. Avasarala, V., Mullen, T., Hall, D.L., Garga, A.: MASM: Market Architecture or Sensor Management in Distributed Sensor Networks. In: SPIE Defense and Security Symposium, Orlando FL, pp. 5813–5830 (2005)
12. Walsh, W.E., Wellman, M., Ygge, F.: Combinatorial Auctions for Supply Chain Formation. In: ACM Conf. on Electronic Commerce, pp. 260–269 (2000)
13. Narahari, Y., Dayama, P.: Combinatorial Auctions for Electronic Business. Sadhana 30(Pt. 2 & 3), 179–211 (2005)