Evolutionary Technique for Combinatorial Reverse Auctions

Shubhashis Kumar Shil, Malek Mouhoub and **Samira Sadaoui**

Department of Computer Science, University of Regina, Regina, Canada
{shil200s, mouhoubm, sadaouis}@uregina.ca

Abstract

Winner(s) determination in combinatorial reverse auctions is a very appealing application in e-commerce but very challenging especially when multiple attributes of multiple instances of items are considered. The difficulty here is to return the optimal solution to this hard optimization problem in a reasonable computation time. In this paper, we make this problem more interesting by considering all-units discounts on attributes and solving it using genetic algorithms. We also consider the availability of instances of items in sellers' stock. In order to evaluate the performance of our proposed method, we conducted several experiments on randomly generated instances. The results clearly demonstrate the efficiency of our method in determining the winner(s) with an optimal procurement cost in an efficient processing time.

Introduction

The main purpose in combinatorial auctions considering multiple items is to increase the efficiency of bid allocation which corresponds to minimizing the procurement cost in a reasonable computation time (Avasarala, Polavarapu, and Mullen 2006; Patodi, Ray, and Jenamani 2011; Zhang 2007) when determining the winner (Gong et al. 2007). Winner determination is an NP-complete problem (Patodi, Ray, and Jenamani 2011; Zhang 2007). It is one of the main challenges in combinatorial auctions (Avasarala, Polavarapu, and Mullen 2006). Combinatorial auctions have been applied to a procurement scenario such as travel packages and transportation (Narahari and Dayama 2005; Rassenti, Smith, and Bulfin 1982). Hsieh and Tsai developed a Langrangian heuristic method to tackle combinatorial auctions (Patodi, Ray, and Jenamani 2011). Ant algorithms have been defined by Sitarz for the same purpose (Patodi, Ray, and Jenamani 2011). Genetic Algorithms (GAs) have been used to solve combinatorial optimization problems (Patodi, Ray, and Jenamani 2011), such as job scheduling (Senthilkumar and Shahabudeen 2006), quadratic assignment problem (Tate and Smith 1995) and travelling salesman problem (Watabe and Kawaoka 2000). In (Shil, Mouhoub, and Sadaoui 2013b), we proposed a GA based method, GACRA, and applied that method to solve winner determination in a Combinatorial Reverse Auctions (CRA) by considering multiple items. In a CRA, there is one buyer and several sellers. The buyer specifies his requirements and the sellers compete to win. In (Shil, Mouhoub, and Sadaoui 2013b), we introduced two repairing methods, RemoveRedundancy and RemoveEmptiness, along with a modified two-point crossover operator. In that paper, we demonstrated that GACRA is better in terms of computation time and procurement cost when compared to another GA-based winner determination method (Patodi, Ray, and Jenamani 2011). In (Shil, Mouhoub, and Sadaoui 2013a), we conducted several statistical experiments and the results showed that GACRA is a consistent method. In (Shil and Mouhoub 2014), we solved CRA by another GA-based method, GAMICRA. In this latter paper, we modified the configurations of chromosomes and the fitness function. In addition, we also improved RemoveRedundancy and RemoveEmptiness to consider multiple instances of items.

In this paper, our target is twofold: (1) to solve the winner determination problem for multiple instances of items in the context of CRA, and (2) to find the winner(s) in a reasonable computation time and reduced procurement cost. In addition to the features of CRA that have been addressed in (Shil and Mouhoub 2014), we have included some new interesting dimensions. First, we consider two attributes, price and delivery rate. We also consider all-units discount strategy (Ebrahim, Razmi, and Haleh 2009) on both price and delivery rate. In our case, the problem is also multi-sourcing as in (Shil and Mouhoub 2014). The buyer can buy different items from different sellers. He can also buy instances of the same item from different sellers. In addition, we consider various situations related to sellers' stock such as the number of available instances of items provided by a given seller is (a) greater than or (b) less than the buyer's requirement or (c) the seller is out of that item. Also, the maximum price constraint of the buyer

as well as the minimum price constraint of the sellers for each item instance is taken into consideration. To tackle these additional features we define the chromosome representation based on the number of items and item instances. We also define the fitness function and keep it simple enough in order to maintain a reasonable computation time. In order to evaluate the time performance of our method to return the best procurement cost, we conducted several experiments on randomly generated instances. Before conducting these experiments, we tuned the different parameters of our proposed method to their best. The results we report in this paper clearly demonstrate that our method is a prominent one. After each round we rank the bids and show the prices to the buyer and the sellers' as well. We also present the break-downs of the number of instances of the items the seller(s) will provide. We perform a statistical analysis to investigate the behavior of our proposed method and show that it does not suffer from the inconsistency issue.

Combinatorial reverse auction is a computationally complex problem and by considering more parameters we add extra dimensions to the complexity. For instance, if there are p items, m instances and n sellers, then the search space is $p \times m \times n$. In addition, the solving procedure needs to satisfy both the buyer's and sellers' constraints which add to the complexity of the problem. Supplier selection is one of the most important in a multi-criteria decision problem (Ebrahim, Razmi, and Haleh 2009; Xia and Wu 2007). In addition, the problem becomes more complex when considering more than one attribute of items and the discount strategy (Xia and Wu 2007).

The rest of this paper is organized as follows. In Section 2, we state the problem we are tackling in details. In Section 3, our proposed method is presented and described through an example. Section 4 reports the experimental study we conducted to evaluate the computation time, procurement cost and the consistency of our method. Finally in Section 5, concluding remarks and future research directions are listed.

Problem Statement

We consider a CRA with multi-attributes of multiple instances of multiple items. To determine the winner we consider the items price along with the delivery rates and the respective discounts. Figure 1 illustrates a sample scenario with the buyer's request and sellers' stock. Here, the buyer requests two instances of item 1, three instances of item 2 and two instances of item 3. Then, he specifies the maximum buying price for each instance of item and terminating condition as follows. The maximum price for each instance of item1, item2 and item3 is 500, 700, and 200 respectively and the maximum number of round is 5.

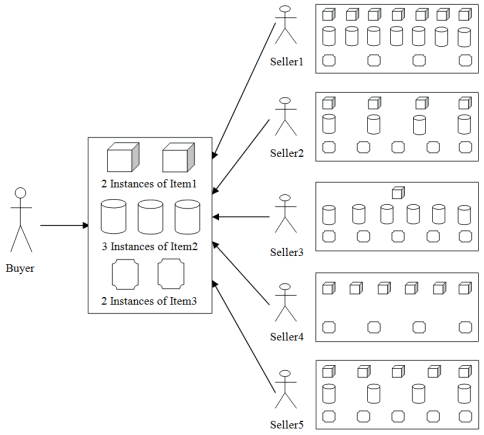

Figure 1: A sample scenario in CRAs.

These are treated as the buyer's constraints. We consider the minimum price the sellers can afford for each instance of items as shown in Table 1. Hence, there is an implicit constraint that the bid price should be greater than or equal to the Minimum Price (MP) of the seller and less than or equal to the buyer's maximum price. Table 1 also shows the Delivery Rate (DR).

Table 1: Sellers' Minimum Price (MP) and Delivery Rate (DR)

	Item1		Item2		Item3	
	MP	**DR**	**MP**	**DR**	**MP**	**DR**
S1	400	40	600	60	120	10
S2	350	35	650	65	100	10
S3	380	35	580	55	150	15
S4	350	55	-	-	150	15
S5	400	35	650	65	120	10

Table 2: Sellers' all-units discount on bid price

	Discount Rate					
	Item1		Item2		Item3	
	>2	**>5**	**>2**	**>5**	**>2**	**>5**
S1	5%	10%	4%	9%	5%	10%
S2	5%	9%	5%	8%	5%	9%
S3	-	-	5%	9%	5%	10%
S4	5%	9%	-	-	4%	10%
S5	5%	10%	5%	10%	4%	9%

Table 3: Sellers' all-units discount on delivery rate

	Discount Rate					
	Item1		Item2		Item3	
	>2	**>5**	**>2**	**>5**	**>2**	**>5**
S1	50%	100%	40%	90%	50%	100%
S2	50%	90%	50%	80%	50%	90%
S3	-	-	50%	90%	50%	100%
S4	50%	90%	-	-	40%	100%
S5	50%	100%	50%	100%	40%	90%

We use all-units discount strategy. The sellers provide discounts on price and delivery rate under some conditions. Tables 2 and 3 show the discount rates on bid price and delivery rate respectively. Given all these data, the goal is to determine the winner(s) with optimal price in a reasonable computation time.

Proposed Method

We discuss our proposed method using the sample scenario of Figure 1. We generate the required number of bits (*rb)* to represent each seller's item instance using the following rule: $n \leq 2^{rb}$ (1)
where n is the number of sellers. Here $n = 5$ and $rb = 3$.

Algorithm 1: Proposed Method (p: number of items, m_p: number of instances of items p specified by the buyer, s_p: number of available instances of items p possessed by sellers, n: number of sellers, δ: number of generations, γ: number of rounds, α: crossover rate, β: mutation rate, ζ: number of chromosomes, *maxPrice*: set of maximum prices of instances of items specified by the buyer, *minPrice*: set of minimum prices of instances of items provided by sellers, d: set of delivery rate, discount rate on bid price and delivery rate)

1. $r = 1$;
   do{
2. $t = 1$;
3. bidGenerator(*maxPrice*, *minPrice*, p, n);
   //checks price constraints and generates bid prices,
   //*bidPrice*
4. chromosomeGenerator(m_p, s_p, p, n, ζ);
   //generates feasible initial chromosomes
5. fitnessOperation(X_ζ: set of initial chromosomes, *price*: calculated from *bidPrice* and d);
   //computes fitness values of chromosomes, *fv*
   do{
6. selectionOperation(X_ζ, *fv*);
   //selects chromosomes using gambling-wheel disk
   //method
7. crossoverOperation(X_ζ^s: set of chromosomes after selection, *fv*);
   //generates child chromosomes from parent
   //chromosomes with modified two-point crossover
   //considering crossover rate, α
8. mutationOperation(X_ζ^c: set of chromosomes after crossover, *fv*);
   //mutates chromosomes considering mutation rate, β
9. newChromosomeGenerator(X_ζ, NX_ζ: set of chromosomes after mutation);
   //removes duplicity and selects better chromosomes from
   //both initial and new chromosomes of generation, δ
   }(while t ≤ δ);
   }(while r ≤ γ);
10. return winner(s);
    //returns winner(s) with minimum bid price

Figure 2: Proposed solving algorithm.

Assume p = the number of items and m_P = the number of instances of $item_P$ where $1 \leq P \leq p$, then the length of chromosome is calculated by the following equation:

$$lengthChromosome = \sum_{P=1}^{p} m_P \times rb \quad (2)$$

Figure 2 shows our proposed GA-based solving algorithm. The procedures bidGenerator, chromosomeGenerator and newChromosomeGenerator are presented in figures 3-5. Here, the number of chromosomes is equal to 4. Table 4 presents the first round bid price and table 5 shows the randomly generated initial chromosome representation. For each instance, a random number from [1, n] is generated and its correspondence binary representation according to rb becomes the part of a chromosome.

Table 4: Bid price generation

Bid Price of Each Instance	S1	S2	S3	S4	S5
Item1	480	450	460	430	490
Item2	670	680	660	-	660
Item3	180	190	170	180	150

According to X1 in table 5, both seller 1 and seller 2 will supply a single instance of both item 1 and item 3. Seller 2 will supply all three instances of item 2. X1 and X4 are feasible but X2 and X3 are not. In X2, seller 3 is selected for two instances of item 1 whereas he has only one instance in his stock. In X3, seller 4 is selected for one instance of item 2 whereas he has no instance of item 2 in his stock. Assume the following valid chromosomes are generated after removing infeasibility as shown in Algorithm 3:

X1: 001 010 010 010 010 001 010
X2: 011 011 101 011 001 101 100
X3: 010 011 010 101 101 011 011
X4: 101 101 001 001 001 001 001

Table 5: Initial chromosomes

Chromosome	Item1		Item2			Item3	
	Ins1	Ins2	Ins1	Ins2	Ins3	Ins1	Ins2
X1	001	010	010	010	010	001	010
X2	011	011	101	011	001	101	100
X3	010	011	100	101	101	011	011
X4	101	101	001	001	001	001	001

To evaluate the fitness value of each chromosome, the following fitness function is applied.

$$F(X_i) = 1 \Big/ \sum_{N=1}^{n} \sum_{P=1}^{p} b_{NP} \times l_{NP} \quad (3)$$

where l_{NP} is the number of instances of item P for seller N and b_{NP} is the price (discounted bid price + discounted delivery rate) of item P submitted by seller N.

Gambling Wheel Disk (Gong et al. 2007) method is used as a selection strategy. After the selection process, the following chromosomes are selected based on the fitness values according to equation (3).

X1: 001 010 010 010 010 001 010 (previous X1)

X2: 011 011 101 011 001 101 100 (previous X2)
X3: 101 101 001 001 001 001 001 (previous X4)
X4: 101 101 001 001 001 001 001 (previous X4)

The modified two-point crossover (Shil, Mouhoub, and Sadaoui 2013b) is then applied as shown in figure 6(a). After crossover, the following chromosomes are generated.

X1: 001 010 101 011 001 001 010
X2: 101 100 010 010 010 011 011
X3: 101 101 001 001 001 001 001
X4: 101 101 001 001 001 001 001

Algorithm 2: bidGenerator (*maxPrice*, *minPrice*, *p*, *n*)

```
bidPrice = +∞;
//initial value before generating bid price in the first round
1.  for each n
2.  {   for each p
3.      {   if(maxPrice > bidPrice)
4.              maxPrice = bidPrice;
5.          generateBid(maxPrice, minPrice);
            //generates bid price less than or equal to
        maxPrice and greater than or equal to minPrice
6.      }
7.  }
```

Figure 3: bidGenerator function.

Algorithm 3: chromosomeGenerator (m_p, s_p, p, n, ζ)

```
1.  for each ζ
2.  {   for each p
3.      {   for each m_p
4.          {   generateRN(n);
                //generates random number between
                //1 and n
5.          }
6.      }
7.      buildChromosome(nm_p: set of random numbers);
8.      //generates chromosome
9.      checkFeasibility(m_p, s_p);
10.     //makes the chromosome feasible
11. }
```

Figure 4: chromosomeGenerator function.

Algorithm 4: newChromosomeGenerator (X_ζ, NX_ζ, ζ)

```
1.  X_ζ ∪ NX_ζ;
2.  uniqueCheck(X_ζ, NX_ζ);
    //returns only the unique chromosomes and xn, the
    //number of unique chromosomes
3.  if(xn < ζ)
4.  { injectChromosome();
    //generates the remaining new chromosomes by
    //invoking chromosomeGenerator()
    }
5.  if(xn > ζ and xn < 2ζ)
6.  { removeChromosome();
    //removes additional chromosomes according to
    //worse fitness values
7.  }
8.  return ζ chromosomes according to better fitness
    values
```

Figure 5: newChromosomeGenerator function.

The mutation is then performed as shown in figure 6(b). Table 6 shows the result of the first generation for round 1. According to this result, seller 1 will provide two instances of item 1 and three instances of item 2 while seller 5 will provide two instances of item 3. Algorithm 1 is repeated until the specified number of generations and rounds are completed.

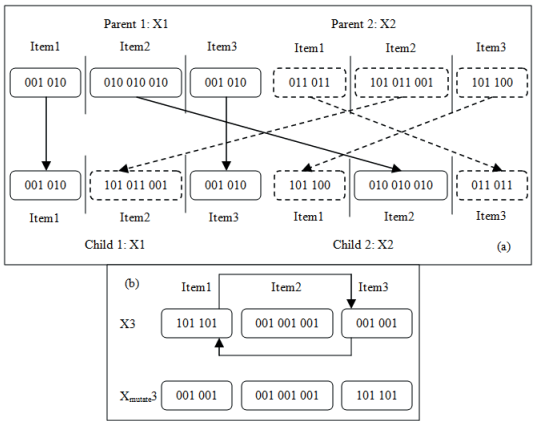

Figure 6: a) Modified two-point crossover b) Mutation.

Table 6: Bid result

Rank	Chromosome	Item 1	Item 2	Item 3	Price
1	X3	S1(2)	S1(3)	S5(2)	3398
2	X2	S4(1), S5(1)	S2(3)	S3(2)	3416
3	X4	S5(2)	S1(3)	S1(2)	3468
4	X1	S1(1), S2(1)	S1(1), S3(1), S5(1)	S1(1), S2(1)	3565

Experimentation

Our proposed method has been implemented in Java and is executed on an Intel (R) Core (TM) i3-2330M CPU with 4 GB of RAM and 2.20 GHz of processor speed. The experiments are conducted on randomly generated instances. *maxPrice* is generated randomly from [100, 1000] whereas *minPrice* is generated from [50% of *maxPrice*, 75% of *maxPrice*]. For delivery rate we use a value from [10% of *minPrice*, 25% of *minPrice*]. 3% to 5% discount on price is considered if more than 10 instances of a given item are supplied by a seller and 6% to 10% when it is more than 25. We also consider 30% to 50% discounts on delivery rate when the seller supplies more than 10 instances of an item and 60% to 100% if it is more than 25. Values of s_p are generated from [0, 30].

Experiment 1: Parameter Tuning

The goal of this first experiment is to tune the parameters of our method to their best values for the specific instances we are using. We performed 27 tests by varying the number of chromosomes from 50 to 200, the crossover rate

from 0.5 to 0.7 and the mutation rate from 0.01 to 0.1. For this experiment, we use the following parameters and settings: number of sellers = 100; number of items = 10; number of instances from [1-100]; number of generations = 50 and number of rounds = 5. From the results of this experiment that are the average of 20 runs, we have decided to use the configuration of test21 for the next experiments as with this configuration, the algorithm performs the best. The configuration of test21 is: number of chromosome = 200, crossover rate = 0.5 and mutation rate = 0.1.

Experiment 2: Statistical Analysis

Figure 7 illustrates the statistical analysis of the proposed method. It depicts the average price with maximum and minimum values of generations. It also shows the error bars with a confidence level of 95%. From this figure it is clearly notable that the method is able to control the solution variations and stabilizes after some generations. Notice that the minimum bid price remains constant after 70 generations which means that the best solution found by the method might be the optimal solution. The results shown in this experiment are the average values of 20 runs. We also performed the same experiment for 50 runs but found no major improvement. For this reason we continue the next experiments for 20 runs and analyze the results based on the average values.

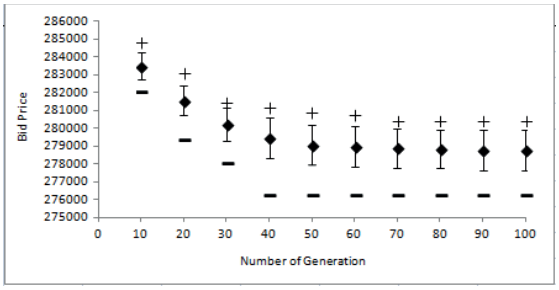

Figure 7: Statistical analysis of the proposed method.

Experiment 3: Number of Generations

We use the following settings for experiments 3 and 4. The maximum number of generations and rounds are 50 and 5 respectively. The number of sellers and items are 100 and 10 respectively. The range of instances is from 1 to 100. Figure 8 reports the results of experiment 3. Figure 8.a shows how the bid price improves with the increase of the number of generations. As we can easily see the cost significantly improves in the first generations and stabilizes afterwards. Figure 8.b shows how the computation time varies with the increase of the number of generations.

Experiment 4: Number of Sellers and Items

Figure 9 shows how the computation time increases when we increase the number of sellers as well as the number of items. In this regard our method is capable of producing the solutions in a very short time even for large number of sellers and items.

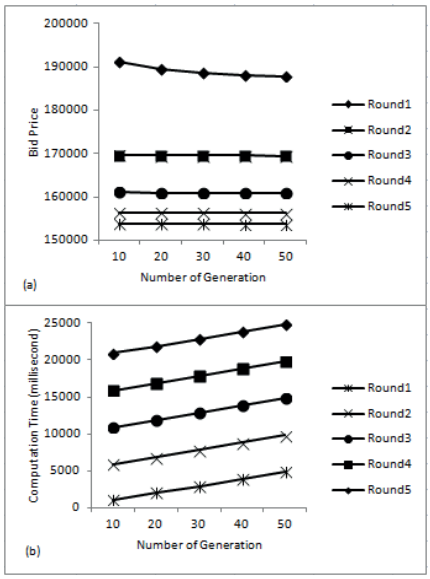

Figure 8: a)Bid price vs number of generations b) Computation time vs number of generations.

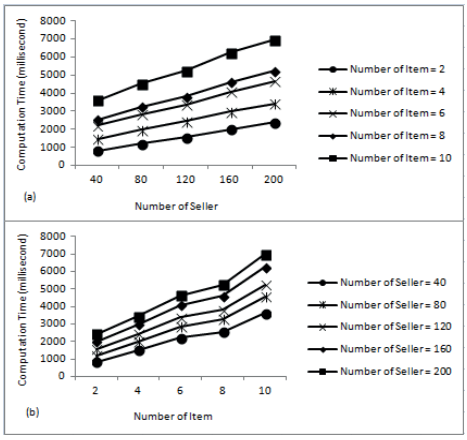

Figure 9: a) Computation time vs number of sellers b)Computation time vs number of items.

Conclusion and Future Work

E-commerce and resource allocation problems need real-time response (Hoos and Boutilier 2000). In these problems, the instances are large and the solutions are needed in a very short time. While the exact algorithms guarantee to return the optimal solution, they often suffer from their exponential time cost especially for larger instances (Hoos and Boutilier 2000). A good alternative is the approximation methods that still produce high-quality solutions in practice with a reasonable computation time (Hoos and Boutilier 2000). For this reason, we have opted for evolutionary techniques to solve the winner

determination problem as they have the ability to produce good solutions from the very first generations. We propose a GA based method for winner determination with two attributes of multiple instances of items in CRAs considering all-units discount on both of these attributes. We show that our proposed method can produce optimal solutions in a reasonable computation time. Moreover, it shows that the method is a consistent one as it is able to reduce the solution variations over different runs.

In the near future, we will apply an exact method to get the optimal solution for the instances we used in the experiments. This will help us evaluate the quality of the solutions (procurement cost) returned by our proposed method. We will also compare the time performance of our method to the well known techniques used to solve related problems (Gonen and Lehmann 2000; Qian et al. 2014). In this regard, we have to acknowledge that we tackle unique instances that have not been solved before. Since parallel GAs are capable of providing the solutions in a better computation time especially when some variable ordering heuristics are used (Abbasian and Mouhoub 2011; Abbasian and Mouhoub 2013; Mouhoub and Jafari 2011), one of the future works is to use Parallel GAs. Another promising direction is to consider, in addition to 'price' and 'delivery rate', more attributes such as 'delivery time', 'seller reputation', and 'warranty'. We will also investigate the applicability of other meta-heuristics together with Gas (Ostler and Wilke 2014). Finally we will study several diversity models while using GAs to find the most appropriate one for our problem (Gupta and Ghafir 2012).

References

Abbasian, R., and Mouhoub, M. 2011. An Efficient Hierarchical Parallel Genetic Algorithm for Graph Coloring Problem. In *Proceedings of the 13th Annual GECCO*, 521–528.

Abbasian, R., and Mouhoub, M. 2013. A Hierarchical Parallel Genetic Approach for the Graph Coloring Problem. *Applied Intelligence,* 39(3): 510-528. Springer.

Avasarala, V., Polavarapu, H., and Mullen, T. 2006. An Approximate Algorithm for Resource Allocation using Combinatorial Auctions. In *Proceedings of the International Conference on Intelligent Agent Technology*, 571–578.

Ebrahim, R. M., Razmi, J., and Haleh, H. 2009. Scatter search algorithm for supplier selection and order lot sizing under multiple price discount environment. *Advances in Engineering Software* 40(9): 766-776.

Gonen, R, and Lehmann, D. 2000. Optimal solutions for multi-unit combinatorial auctions: Branch and bound heuristics. In *Proceedings of the 2nd ACM conference on Electronic commerce*, 13-20.

Gong, J., Qi, J., Xiong, G., Chen, H., and Huang, W. 2007. A GA Based Combinatorial Auction Algorithm for Multi-robot Cooperative Hunting. In *Proceedings of the International Conference on Computational Intelligence and Security*, 137–141.

Gupta, D., and Ghafir, S. 2012. An overview of methods maintaining diversity in genetic algorithms. *International Journal of Emerging Technology and Advanced Engineering* 2(5): 56-60.

Hoos, H. H., and Boutilier, C. 2000. Solving Combinatorial Auctions using Stochastic Local Search. In *Proceedings of the 17th National Conference on Artificial Intelligence*, 22–29.

Mouhoub, M., and Jafari, B. 2011. Heuristic techniques for variable and value ordering in CSPs. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, 457-464.

Narahari, Y., and Dayama, P. 2005. Combinatorial Auctions for Electronic Business. *Sadhana* 30 (pt. 2 & 3): 179–211.

Ostler, J., and Wilke, P. 2014. Improvement by combination how to increase the performance of optimisation algorithms by combining them. In *Proceedings of the 10th international conference of the practice and theory of automated time tabling*, 359-365.

Patodi, P., Ray, A.K., and Jenamani, M. 2011. GA Based Winner Determination in Combinatorial Reverse Auction. In *Proceedings of the 2nd International Conference on Emerging Applications of Information Technology (EAIT)*, 361–364.

Qian, X., Huang, M., Gao, T., and Wang, X. 2014. An improved ant colony algorithm for winner determination in multi-attribute combinatorial reverse auction. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC)*, 1917-1921.

Rassenti, S. J., Smith, V. L., and Bulfin, R. L. 1982. A Combinatorial Auction Mechanism for Airport Time Slot Allocation. *The Bell Journal of Economics* 13: 402–417.

Senthilkumar, P., and Shahabudeen, P. 2006. GA Based Heuristic for the Open Job Scheduling Problem. *International Journal of Advanced Manufacturing Technology* 30: 297–301.

Shil, S. K., and Mouhoub, M. 2014. Considering Multiple Instances of Items in Combinatorial Reverse Auctions. In *Proceedings of the 27h International Conference on Industrial, Engineering & Other Applications of Applied Intelligent Systems (IEA/AIE),* 487-496. Springer.

Shil, S. K., Mouhoub, M., and Sadaoui, S. 2013a. An Approach to Solve Winner Determination in Combinatorial Reverse Auctions Using Genetic Algorithms. In *Proceedings of the 15th Annual GECCO*, 75–76.

Shil, S. K., Mouhoub, M., and Sadaoui, S. 2013b. Winner Determination in Combinatorial Reverse Auctions. In *Proceedings of the 26th International Conference on Industrial, Engineering & Other Applications of Applied Intelligent Systems (IEA/AIE)*, 35–40. Springer.

Tate, D. M., Smith, A. E. 1995. A Genetic Approach to the Quadratic Assignment Problem. *Computers and Operations Research* 22(1): 73–83.

Watabe, H., and Kawaoka, T. 2000. Application of Multi-Step GA to the Traveling Salesman Problem. In *Proceedings of the 4th International Conference on Knowledge-Based Intelligent Engineering Systems and Allied Technologies* 2: 510-513.

Xia, W., and Wu, Z. 2007. Supplier selection with multiple criteria in volume discount environments. *Omega* 35(5): 494-504.

Zhang, L. 2007. The Winner Determination Approach of Combinatorial Auctions based on Double Layer Orthogonal Multi-Agent Genetic Algorithm. In Proceedings of the 2nd IEEE Conference on Industrial Electronics and Applications, 2382–2386.