Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2015, Article ID 639787, 10 pages
http://dx.doi.org/10.1155/2015/639787

Research Article

A New Hybrid Algorithm to Solve Winner Determination Problem in Multiunit Double Internet Auction

Mourad Ykhlef[1] and Reem Alqifari[2]

[1]*Department of Information Systems, College of Computer and Information Sciences, King Saud University, P.O. Box 51178, Riyadh 11543, Saudi Arabia*
[2]*Information Technology Department, College of Computer and Information Sciences, P.O. Box 51178, Riyadh 11543, Saudi Arabia*

Correspondence should be addressed to Mourad Ykhlef; ykhlef@ksu.edu.sa

Received 3 November 2014; Revised 1 March 2015; Accepted 1 March 2015

Academic Editor: Pandian Vasant

Solving winner determination problem in multiunit double auction has become an important E-business task. The main issue in double auction is to improve the reward in order to match the ideal prices and quantity and make the best profit for sellers and buyers according to their bids and predefined quantities. There are many algorithms introduced for solving winner in multiunit double auction. Conventional algorithms can find the optimal solution but they take a long time, particularly when they are applied to large dataset. Nowadays, some evolutionary algorithms, such as particle swarm optimization and genetic algorithm, were proposed and have been applied. In order to improve the speed of evolutionary algorithms convergence, we will propose a new kind of hybrid evolutionary algorithm that combines genetic algorithm (GA) with particle swarm optimization (PSO) to solve winner determination problem in multiunit double auction; we will refer to this algorithm as AUC-GAPSO.

1. Introduction

Internet auctions appeared on the scene in the mid-1990s and quickly became one of the most successful applications of electronic commerce [1]. Auction is defined as a market mechanism for accepting bids or offers from buyers or sellers and then used a set of rules to allocate goods. Double auction includes multiple sellers and multiple buyers in the same market where they are competing against each other for a variety of different items [2, 3]. In a multiunit double auction, each participant should determine how many units of an item and in which price he/she asks or bids. When a buyer's bid exceeds or matches a seller's ask, a trade will occur [4]. The most important issue in a double auction is how to allocate or match buyer with seller to maximize profit in the market. This problem is known as winner determination problem [5].

There are two approaches to solve this problem: exact approach and evolutionary approach; the exact approach, such as linear programming or simplex method, gives the optimal solution but in exponential time while evolutionary approach, such as genetic algorithm (GA) [2, 6] and particle swarm optimization (PSO) [3], solves the problem in a short time without the guarantee of obtaining always the optimal solution. In [7] a parallel genetic algorithm hybridization with local search was proposed. Other metaheuristics have been also used in double auction like memetic algorithm, Tabu search; in [8] "a computational experience regarding four well-known metaheuristics (stochastic local search, Tabu search, genetic algorithms, and memetic algorithms) for solving the winner determination problem" has been conducted.

Genetic algorithm is a general purpose search algorithm which uses principles inspired by natural genetic populations [9]. Particle swarm optimization (PSO) can be considered as an alternative to the standard GAs. The PSO was inspired by insect swarms and has been shown to be a competitor to the GA for optimization problems. Since then, several improved PSO general purpose algorithms have been developed [10, 11]. Both GA and PSO algorithms have shown good performance for some particular applications but not for other ones. For example, sometimes GAs outperformed PSO, but occasionally the opposite happened showing the typical application

driven characteristic of any single technique. This is due to the different search method adopted by the two algorithms, the typical selection-crossover-mutation approach versus the velocity-position-global-local best communication [12–15].

In order to improve the speed of convergence of existing multievolutionary algorithms, we will propose a new kind of hybrid evolutionary algorithm AUC-GAPSO that consists of combination of GA and PSO to solve winner determination problem in a multiunit double auction. AUC-GAPSO focuses on decreasing the time with good solution.

The rest of the paper is organized as follows. Double auction is defined in Section 2; however genetic algorithm and particle swarm optimization are defined in Section 3. Solving winner determination problem in double auction using our AUC-GAPSO algorithm is detailed in Section 4. Experimental results are reported in Section 5. We conclude this work in Section 6.

2. Double Auction

Double auction is referring to multiple buyers and sellers competing among each other. The double auction is divided into single-unit double auction and multiunit double auction [16]. In multiunit double auctions there are many sellers' bids prices and many buyers' bids for a variety of items with multiunit. Trade occurs when the buyers bid is greater than the sellers bids [2, 3, 17, 18]. Both the seller and the buyer have to determine the price and quantity of each specific item as follows:

$$\begin{aligned} &\text{Buyer} \longrightarrow \text{item, price, and quantity} \\ &\text{Seller} \longrightarrow \text{item, price, and quantity.} \end{aligned} \tag{1}$$

The auctioneer assigns buyers to sellers based on their preferences; there are many sellers asks that may satisfy some buyers' bids and vice versa. There are two kinds of double auctions: synchronous double auction (SDA) and asynchronous double auction (ASDA). Discrete or continuous time of the trading process is the main difference between SDA and ASDA. In the continuous double auction when the auctioneer finds compatible bids, it will match buyers and sellers and clear the market. On the contrary, the auctioneer in the synchronous auction gets all bids in a preset period of time and then makes the match and clears the market [2]. The factors that play roles in winners' determination can be listed as follows [19].

(i) Aggregation: the market maker role in auction is to dissemble or reassemble boundless of items. The options include buy-side, sell-side aggregation or it may include both sides' aggregation. However, if aggregation is not permitted, each bid and ask needs to be matched separately.

(ii) Divisibility: a divisible bid means that bidders can accept a trade of part of the quantity that they have specified at bidding while the indivisible bid means that bidders will reject the bid if the quantity is less than what they bid for [20].

(iii) There are homogeneous/heterogeneous goods.

This paper focuses on two features of the auction which are price and quantity; we will not include other performance factors such as quality, warranty, shipping time, and cost. We are particularly interested in solving winner determination problem in a multiunit synchronous double auction using a hybridization of genetic algorithm and particle swarm optimization.

3. Genetic Algorithm and Particle Swarm Optimization

Genetic algorithm was inspired by Darwin's theory about evolution [9]. Genetic algorithm can be defined as a particular class of evolutionary algorithms that uses techniques inspired by evolutionary biology such as inheritance, mutation, selection, and crossover. Genetic algorithm starts with a set of random solutions represented as chromosomes which are called population. The term chromosome usually refers to a candidate solution for a problem that is often encoded as a bit string. The "gene" is a part of chromosome. Crossover typically consists of exchanging genes between two single chromosomes (parents). Mutation is usually flipping the bit at randomly chosen genes. Forming a new population is driven by the previous populations using crossover and mutation. Hoping that, the new population will be better than the old one. Solutions which are selected to form new solutions (offspring) are selected according to their fitness; chances of reproducing are linked by their suitability. This is repeated until stop condition (e.g., number of populations or improvement of the best solution) is reached. Mostly, the genetic algorithm requires a fitness function which assigns a score to each chromosome in the current population [9].

Particle swarm optimization (PSO) was originally proposed by Kennedy and Eberhart in 1995 [21]. Social behavior and movement dynamics of insects, birds, and fish were the main inspiration. The PSO algorithm is derived from a simplified social model that is closely linked to the swarming theory [22]. In PSO, individual particles of a swarm represent potential solutions. Each particle moves through the problem search space looking for an optimal or acceptable solution. The particles current positions are broadcasted to their neighboring particles. Velocity and the difference between its current positions (the best position found by its neighbors and the best position it has found so far) are the controllers of updating the position of each particle [23].

A good comparison between PSO and GA can be found in [14] where the authors proved that the difference in computational cost between PSO and the GA is problem dependent. The authors of [14] also conclude that PSO outperforms the GA with a greater degree of difference in computational cost when used to solve unconstrained nonlinear problems with continuous variables and smaller when applied to constrained nonlinear problems with continuous or discrete variables. GA has been widely applied to solve complex optimization problems because it can control both discrete and continuous variables, in addition to nonlinear objective and constraint functions [24]. In contrast, PSO may face a problem with a constrained optimization problem [11].

However, both GA and PSO have strengths and weaknesses; hybridization of GA and PSO could lead to further advances that have been proved by [11–13, 15, 25]. In addition, the hybridization could be achieved by different ways or structure.

In this paper, a hybridization of GA with PSO, called AUC-GAPSO, is proposed to overcome the limitations of GA and PSO and to combine their advantages. The convergence in GA has been controlled through crossover and mutation rates. In fact the decrease of inertia weight increases the swarm's convergence. The main problem with PSO is that it prematurely converges [10] to stable point which can be avoided through the genetic operator (crossover and mutation).

4. Solving Winner Determination Problem in Double Auction Using AUC-GAPSO Algorithm

In this section, we describe the AUC-GAPSO algorithm for solving winner determination problem in multiunit double auction. We first explain how we represented the auction dataset in our formalism, and then we present our chromosome structure, encoding scheme, and particles. After that, we describe genetic operators and define the utility or fitness assignment and selection criteria. Finally, we give the algorithmic structure and the pseudocode of AUC-GAPSO algorithm.

4.1. Dataset. Dataset contains data which AUC-GAPSO algorithm is applied to, in order to match bidders. The sellers/buyers information contains bidder ID, item, max quantity, maximum price that the buyers are willing to pay or the minimum price that the sellers are willing to accept, and sensitivity to price and quantity. Table 1 gives a sample of all possible matches of buyers and sellers; however Table 2 gives possible matches' matrix that can be derived from Table 1.

4.2. Chromosome. The chromosome in AUC-GAPSO represents all the winners with price and quantity of their winning bids. Also, in the chromosome we should match sellers with buyers. The size of chromosome is determined based on number of sellers, buyers, and items. Consequently, number of possible matches should be calculated to be equal to number of buyers multiplied by number of sellers for each item they bid/ask for.

There are many types of encoding that we can choose based on the problem such as binary encodings [26], which are the most common type; many characters; real-valued encodings [20]; and integer encoding [9]. Chromosomes representation became an issue for the different auction problems based on the auction type. There are many alternatives to represent a chromosome based on other problem domains. To decide which one is the best representation to solve double auction problem, Goldberg in his 1989 textbook recommends that "*the user should select a coding so that short, low-order schemata are relevant to the underlying problem and relatively unrelated to schemata over other fixed positions*" and "*the user should select the smallest alphabet that permits a natural expression of the problem*" [27]. The main meaning behind these recommendations is that the appropriate choice of genetic representation depends on the problem. According to that, in our problem, we will choose real-valued representation because it is the suitable one for constrained optimization problem. The sample of Tables 1 and 2 is represented by the chromosome of Table 3 where each couple of genes (price P and quantity Q) belongs to one match of the possible matches' matrix (Table 2); for example, buyer 1 and seller 1 won item 1 for price 13 and quantity 6.

Moreover, it is important to obtain sellers asks and buyers bids. In order to have less space, that information in addition to item vector will be kept as datasets.

4.3. Particles. After representing the chromosomes, each chromosome has to be linked with a particle that includes PSO variables related to position and velocity. Chromosomes can update their position and velocity using

$$V_{i+1} = \left[(w * V_i) + (c_1 * r_1) \otimes (p_i - x_i) + (c_2 * r_2) \otimes \left(p_g - x_i\right)\right], \quad (2)$$

$$x_{i+1} = x_i + V_{i+1}. \quad (3)$$

Table 1: Bidders information.

Bidder	Item	Quantity	Price	Price sensitivity	Quantity sensitivity
b1	1	10	14	2	1
b2	2	3	15	2	2
b3	1	5	12	1	1
s1	1	9	10	2	1
s2	2	3	14	2	1
s3	1	6	11	2	1

Table 2: Possible matches matrix.

Buyer	Seller	Item
1	1	1
1	3	1
2	2	2
3	1	1
3	3	1

Table 3: Chromosome.

P	Q	P	Q	P	Q	P	Q	P	Q
13	6	12	4	15	3	10	3	11	2

Particles local and global best position can be updated by using

$$p_{i+1} = \begin{cases} p_i & \text{if } p_i \geq x_{i+1} \\ x_{i+1} & \text{if } p_i < x_{i+1}, \end{cases} \quad (4)$$

$$p_g = \begin{cases} p_g & \text{if } p_g \geq p_i \\ p_i & \text{if } p_g < p_i, \end{cases} \quad (5)$$

where $i = [1 \cdots \text{MaxPopulation}]$ and w is called inertia weight which controls the influence of previous velocity v_i on the new velocity v_{i+1}. Variable v_i denotes the velocity of the ith particle in the swarm, x_i denotes particle position, p_i denotes the personal best position, p_g denotes the best position found by particles in its neighborhood, c_1 and c_2 are acceleration coefficients, and r_1; r_2 are random numbers, uniformly distributed in $[0 \cdots 1]$ which are used to maintain the diversity of the population. The symbol $\otimes$ denotes pointwise vector multiplication. The population size refers to number of particles in the iterations. As in the genetic algorithm, the initial populations are generated randomly. Eberhart and Shi proved that the performance of the PSO is not sensitive to the population size [12]. The maximum generations refer to the maximum number of generations allowed for the utility value to converge with the optimal solution [28]. Acceleration coefficients c_1 and c_2 [23, 29, 30] are two positive numbers such that $c_1 + c_2 = 4$; c_1 is a self-confidence factor to determine the relative influence of the cognitive component and c_2 is a swarm confidence factor used to determine the relative influence of the social component.

The PSO algorithm used a constant value for the inertia weight w; it could be equal to 0.9 as proposed by [30]. This inertia weight could also linearly decrease with respect to time [23, 31, 32]. Generally for initial stages of the search process, large inertia weight to enhance the global exploration is recommended while, for last stages, the inertia weight is reduced for local exploration [25, 28, 33]. The mathematical expression could be as follows:

$$\text{Inertia weight } w = w_{\max} - \left(\frac{w_{\max} - w_{\min}}{\text{MaxIter}} \right) + \text{Iter}, \quad (6)$$

where $w_{\max}$ is an initial value of the inertia weight, $w_{\min}$ is a final value of the inertia weight, Iter is a current iteration, and MaxIter is the maximum number of allowable iterations. The values of $w_{\max}$ and $w_{\min}$ that we have been using in our algorithm are 0.9 and 0.4, respectively [23]. We will define the inertia weight w to linearly decrease from 0.9 to 0.4. During the first 5,000 iterations of our algorithm w will be decreased by 0.0001 and then stay constant at 0.4 for the next iterations.

4.4. Genetic Operators. Genetic algorithm uses genetic operators to generate the offspring of the existing population. In this section, we describe three operators of genetic algorithms that we used in our algorithm: selection, crossover, and mutation.

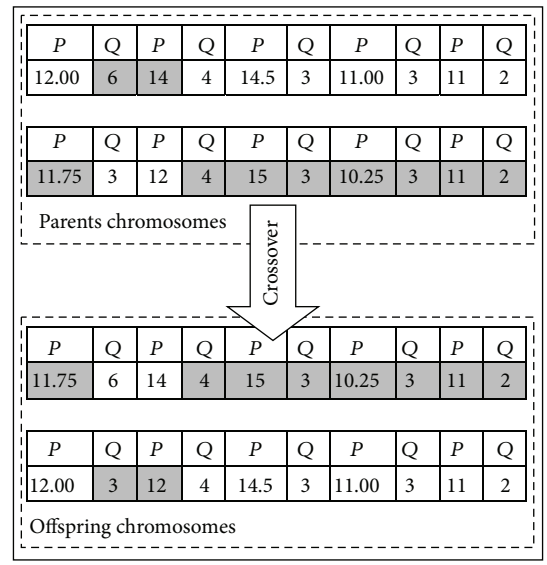

Figure 1: Crossover operator.

4.4.1. Selection. Variety of techniques can be used by a genetic algorithm to select the chromosomes to be copied into the new population. In AUC-GAPSO algorithm, we used "Elitism," where the fittest members of each generation are copied into the next generation.

4.4.2. Crossover. Crossover is working to combine two or more solutions to create new one (i.e., offspring). It is used in a way to come up with possibly better solutions. In most crossover operators, selecting chromosome for crossing over is based on a probability known as crossover probability. In AUC-GAPSO algorithm, we have used uniform crossover operation; crossover can be applied as shown in Figure 1.

4.4.3. Mutation. Mutation is the simplest operator but it is rarely happened. In mutation one or more chromosome genes values are altered randomly. The mutation is used to increase the variability of the population. Also, it ensures that every point in the search space will be reached. There are many mutation operators depending on the problem and the representation type. One of the most common mutations for the binary string is the bit-flip mutation. In contrast, in real-value encoding, the mutation is done by adding or subtracting a small number to the selected genes. The AUC-GAPSO mutation can be applied as illustrated in Figure 2.

4.5. Utility Function. The utility (or fitness according to GA literature) function is used to measure the performance of the chromosomes. In double auction the trade price alone is not ideal or makes the most profit for both sellers and buyers. Here we have the problem where we have to improve the reward of the double auction to match the ideal price and quantity and make the best profit for sellers and buyers according to their bids and predefined quantities. So how can we improve the reward for the auction bidders? It can be done through utility mechanism by improving how much

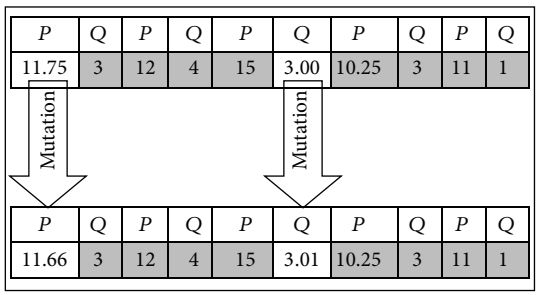

Figure 2: Mutation operator.

bidders can gain in the auction process. The highest utility of the seller when it matches with the highest bid price of buyers, however the highest utility for buyers is reached when he get matched with lowest bid price of sellers. So, to design an efficient auction the utility should be maximized [2]. The most suitable utility function is to maximize sellers and buyers utility function.

The utility function that can be solved as a linear programming problem to find the optimal solution was proposed by [34] where the authors represented the utility as the following linear equation:

$$\begin{aligned}\text{Max} \sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{a=1}^{k}\left(P_{i\cdot j,a} - P_{i,a}^{\text{WILL}}\right)\cdot Q_{i\cdot j,a} \\ + \sum_{j=1}^{n}\sum_{i=1}^{m}\sum_{a=1}^{k}\left(P_{j,a}^{\text{WILL}} - P_{i\cdot j,a}\right)\cdot Q_{i\cdot j,a}.\end{aligned} \tag{7}$$

Consider the following constraints:

$$\begin{aligned}&\sum_{i=1}^{m} Q_{i,a} - \sum_{j=1}^{n} Q_{j,a} = 0 \quad \forall i \in S,\ \forall j \in B,\ \forall a \in \{1,\ldots,k\},\\ &P_{i\cdot j,a} \geq P_{i,a}^{\text{WILL}} \quad \forall i \in S,\ \forall j \in B,\ \forall a \in \{1,\ldots,k\},\\ &P_{i\cdot j,a} \leq P_{j,a}^{\text{WILL}} \quad \forall i \in S,\ \forall j \in B,\ \forall a \in \{1,\ldots,k\},\\ &0 \leq Q_{i,a} \leq Q_{i,a}^{\text{MAX}} \quad \forall i \in (S \cup B),\ \forall a \in \{1,\ldots,k\},\end{aligned} \tag{8}$$

where m is the total number of sellers, n is the total number of buyers, k represents total number of items, $P_{i\cdot j,a}$ is the price of item a traded between seller i and buyer j, $Q_{i\cdot j,a}$ is the quantity of item a traded between seller i and buyer j, $P_{i,a}^{\text{WILL}}$ is seller i's willingness to receive (the minimum acceptable selling price) for item a, $P_{j,a}^{\text{WILL}}$ is buyer i's willingness to pay (the maximum acceptable buying price) for item a, S is set of sellers, in which there are $|S| = m$ sellers, and B is the set of buyers, in which there are $|B| = n$ buyers.

We will use utility function of [2] where each bidder has his own utility function; therefore, he can specify his sensitivity or the range of acceptance of each feature (price and quantity). This utility function is shown as in

$$\begin{aligned}\text{Max} \sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{a=1}^{k}\left(f_1^i\left(P_{i\cdot j,a}\right) - P_{i,a}^{\text{WILL}}\right)\cdot f_2^i\left(Q_{i\cdot j,a}\right) \\ + \sum_{j=1}^{n}\sum_{i=1}^{m}\sum_{a=1}^{k}\left(P_{j,a}^{\text{WILL}} - f_1^i\left(P_{i\cdot j,a}\right)\right)\cdot f_2^i\left(Q_{i\cdot j,a}\right) \\ \forall i \in S, \quad \forall j \in B, \quad \forall a \in \{1,\ldots,k\}.\end{aligned} \tag{9}$$

We can see that the price and the quantity have been replaced with functions. The authors of [2] suggest two types of function the early-growth style and the late-growth style based on buyer/seller choice; early-growth style is concave downward. In contrast, the late-growth style is concave upward. In early-growth style the bidder has most influence in changing price or quantity. In the following $Q_{i,a}^{\text{MAX}}$ means trader i's maximum acceptable quantity for item a; t means the level of steepness ($t = 1, 2, 3, 4, 5$).

Early-growth equations are as follows:

price:

$$\begin{aligned}&f_1^i\left(P_{i\cdot j,a}\right) \\ &= \left\{-\left(\frac{P_{i\cdot j,a} - P_{i,a}^{\text{WILL}}}{P_{j,a}^{\text{WILL}} - P_{i,a}^{\text{WILL}}} - 1\right)^{2t} + 1\right\} \\ &\cdot \left(P_{j,a}^{\text{WILL}} - P_{i,a}^{\text{WILL}}\right) + P_{i,a}^{\text{WILL}} \quad \forall j \in B,\ \forall a \in \{1,\ldots,k\};\end{aligned} \tag{10}$$

seller quantity:

$$f_2^i\left(Q_{i\cdot j,a}\right) = \left\{-\left(\frac{Q_{i\cdot j,a}}{Q_{i,a}^{\text{MAX}}} - 1\right)^{2t} + 1\right\}\cdot Q_{i,a}^{\text{MAX}} \quad \forall i \in S, \quad \forall j \in B, \quad \forall a \in \{1,\ldots,k\}; \tag{11}$$

buyer quantity:

$$f_2^i\left(Q_{i\cdot j,a}\right) = \left\{-\left(\frac{Q_{i\cdot j,a}}{Q_{j,a}^{\text{MAX}}} - 1\right)^{2t} + 1\right\}\cdot Q_{j,a}^{\text{MAX}} \quad \forall i \in S, \quad \forall j \in B, \quad \forall a \in \{1,\ldots,k\}. \tag{12}$$

Late-growth equations are as follows:

price:

$$f_1^i\left(P_{i\cdot j,a}\right)\left(\frac{P_{i\cdot j,a} - P_{i,a}^{\text{WILL}}}{P_{j,a}^{\text{WILL}} - P_{i,a}^{\text{WILL}}} - 1\right)^{2t}\cdot\left(P_{j,a}^{\text{WILL}} - P_{i,a}^{\text{WILL}}\right) + P_{i,a}^{\text{WILL}} \quad \forall j \in B, \quad \forall a \in \{1,\ldots,k\}; \tag{13}$$

seller quantity:

$$f_2^i\left(Q_{i\cdot j,a}\right) = \left(\frac{Q_{i\cdot j,a}}{Q_{i,a}^{\mathrm{MAX}}} - 1\right)^{2t} \cdot Q_{i,a}^{\mathrm{MAX}} \quad \forall i \in S, \quad \forall j \in B, \quad \forall a \in \{1, \ldots, k\}; \tag{14}$$

buyer quantity:

$$f_2^i\left(Q_{i\cdot j,a}\right) = \left(\frac{Q_{i\cdot j,a}}{Q_{j,a}^{\mathrm{MAX}}} - 1\right)^{2t} \cdot Q_{j,a}^{\mathrm{MAX}} \quad \forall i \in S, \quad \forall j \in B, \quad \forall a \in \{1, \ldots, k\}. \tag{15}$$

The advantage of [2] utility is that the participant has the flexibility to change his preferences in each round while in [34] utility the participant has identical utility. The authors of [3] suggested a fuzzy utility function.

4.6. Termination. AUC-GAPSO operations applied for all parents in all iterations. At the end of an operation, new chromosomes (offspring) generated as a result of mutation and crossover operators. These chromosomes should be evaluated by using utility or fitness function. Therefore, AUC-GAPSO algorithm obtained the best N chromosome, where N is the size of population in each generation. This process is repeated until a termination condition is reached. There are many termination conditions that can be used. In AUC-GAPSO the condition is either the maximum generation number (G) is reached or the global best (Pg) is not improved for a specific number of times. At the end, the best chromosome of the last iteration is considered as the fittest matching.

4.7. AUC-GAPSO Algorithm to Solve Winner Determination Problem in Multiunit Double Auction. In this section we present the algorithmic structure of AUC-GAPSO that we produced in addition to the description of its pseudocode. The flowchart of AUC-GAPSO algorithm is shown in Figure 3. After finding the possible matches, the algorithm starts by initializing the population. Then, its fitness or utility value is determined for each chromosome. Subsequently, the following processes are repeated until the prespecified maximum number of generations is achieved or the global best is not improved for a predetermined period of times.

The pseudocode of AUC-GAPSO is shown in Algorithm 1. First, population size, maximum number of generations, and buyers and sellers bidding are taken as input. Then AUC-GAPSO algorithm works as follows.

(1) Loop counter and counter of global best improvement are set to zero.

(2) Find the possible matches from the bidding.

(3) Initial population is generated.

(4) Calculate the fitness (utility) for each chromosome using formula (9) and particle variables (velocity and position are set to zero while local and global best positions are set using formulas (4) and (5), resp.).

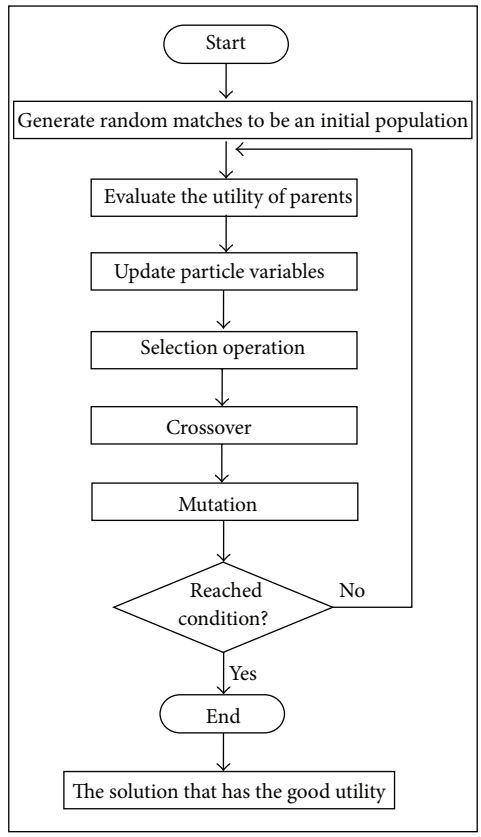

Figure 3: AUC-GAPSO flowchart.

(5) Preserve global best, to be compared to the new one, in order to decide whether it is improved or not.

(6) Selection operation is conducted to find best chromosome in population P and copy the best chromosome in the new population Pn.

(7) Population P is mutated and crossed-over based on the probability of crossover and mutation and then put into Pn.

(8) Calculate Pn fitness and particle variables value using formulas (9) and (2), (3), (4), and (5), respectively.

(9) Update CountPg according to the value of Pg. If the global best did not improve after the selection, between original P and Pn, then the counter will be increased. This step takes advantage of PSO in order to reduce the time.

(10) Increase the loop counter.

(11) Check the termination condition. If the maximum generation number is reached or when the global best is not improved for a specific number of times (i.e., CountPg variable equals quarter maximum generation number), then stop the algorithm and put P into W, which have the fittest matches conducted from the algorithm. Otherwise, go to step (4).

Input:
population size: N
Maximum number of generations: G
Buyers input: B
Sellers input: S
Output:
Interesting Winners: W
Algorithm:
(1) Initialize $t = 0$ and CountPg = 0
(2) Find possible matches from B and S
(3) Generate population P of size N
(4) For each chromosome $i \in P$
  (4.1) Generate winners i from chromosome i
  (4.2) Calculate fitness of winners i
  (4.3) Initialize $X0$, $V0$, Pi, Pg
(5) OldPg = Pg
(6) For each chromosome i and $i + 1 \in P$
  (6.1) Select the best chromosomes to be copied into Pn
  (6.2) Mutate and crossover chromosome i with $i + 1$
  (6.3) Add reproduced chromosome to Pn
  (6.4) Generate winners i from reproduced chromosome i
  (6.5) Calculate fitness of winners i
  (6.6) Update Vi, Xi, Pi, Pg
(7) If OldPg = Pg then
    CountPg ++
  Else
    CountPg = 0
(8) t++
(9) If (($t > G$) or (CountPg > (G/4))) then
    get the fittest chromosome C from P
    $W = C$
    Stop algorithm
  Else
    Go to Step (4)

Algorithm 1: AUC-GAPSO algorithm.

Table 4: AUC-GAPSO output.

Buyer	Seller	Item	Price	Quantity	Fitness
1	1	1	11.08	9	
1	3	1	11.76	1	
2	2	2	15	1	69.4508
3	1	1	10.18	1	
3	3	1	11.02	4	

5. Experiments Results

The programming of AUC-GAPSO algorithm is done by using java Borland JBuilder environment on Intel Core 2 Duo 3 GHz computer with 4 GB RAM, running with Microsoft Windows 7. For controlling the parameters of GA part, the crossover and mutation rates are set at 70% and 30%, respectively. During the experiments, all the bidding, items, population size, and generation were given by the user. The output of the experiments is a file that includes the most suitable solution or matches (fittest chromosome), along with its fitness, for example, for the scenario where there are three buyers, three sellers, and two items (see Tables 1 and 2). The output fittest solution obtained by AUC-GAPSO is given in Table 4.

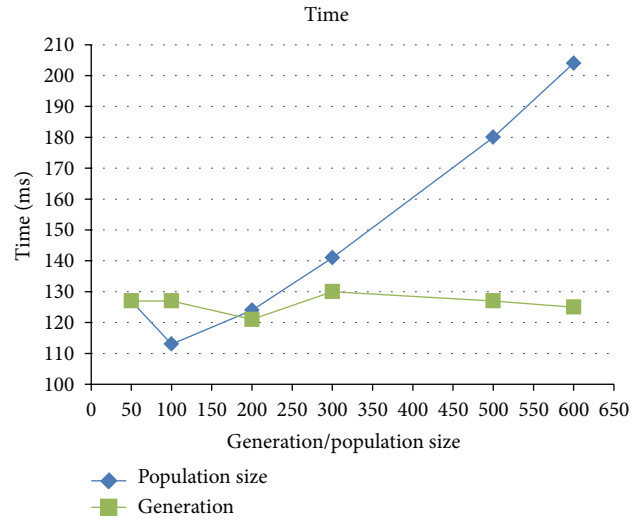

Figure 4: The effect on time by altering population/generation size.

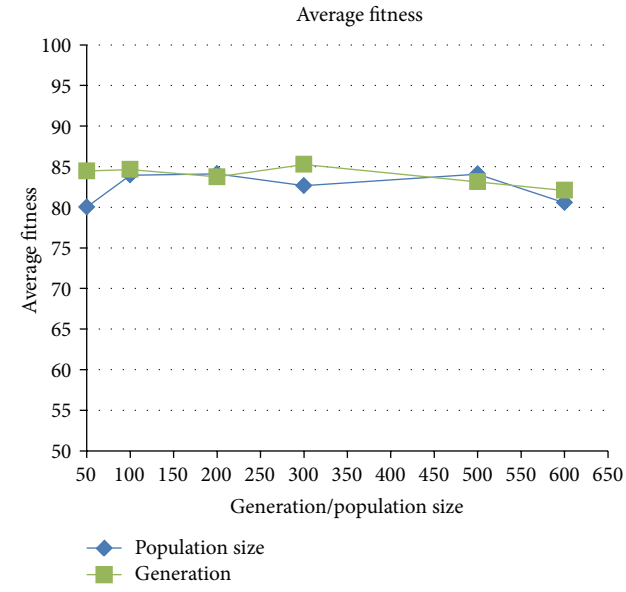

Figure 5: The effect on average fitness by altering population/generation size.

We conducted two experiments for AUC-GAPSO in order to test the speed and fitness or utility. First experiment set the population to 200 and the generation of [50$\cdots$600] while the second experiment set the generation to 200 and the population of [50$\cdots$600]. Referring to the result of AUC-GAPSO of Figure 4, it can be observed that AUC-GAPSO spends less time to be completed. In addition, in first experiment of AUC-GAPSO, increasing of the generation will not necessarily affect the increase of the spent time because its termination condition does not only depend on the generation (iteration), but also depends on the improvement

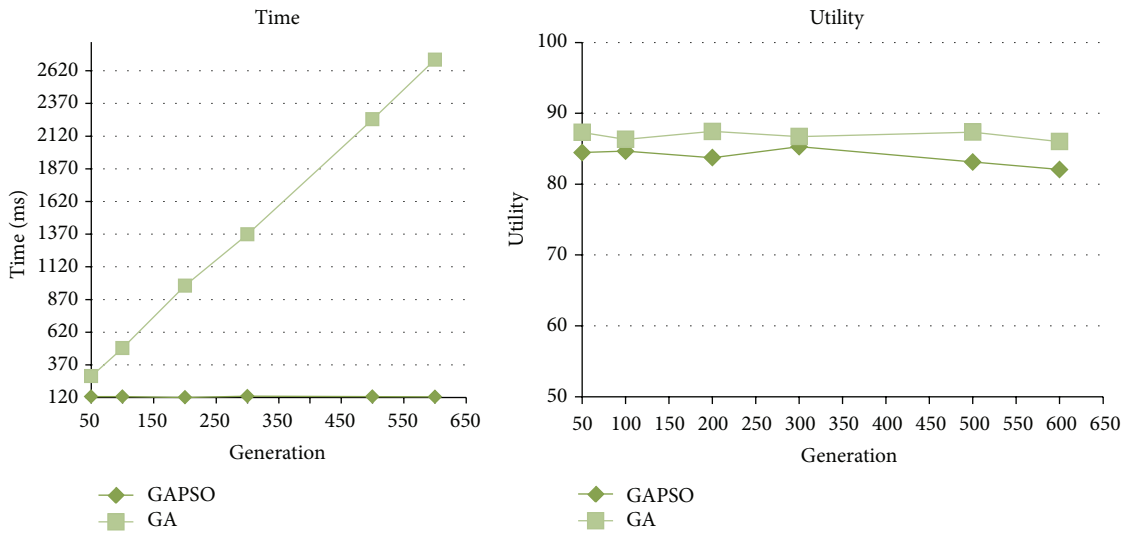

Figure 6: Comparison between GA and GAPSO on effect of generation.

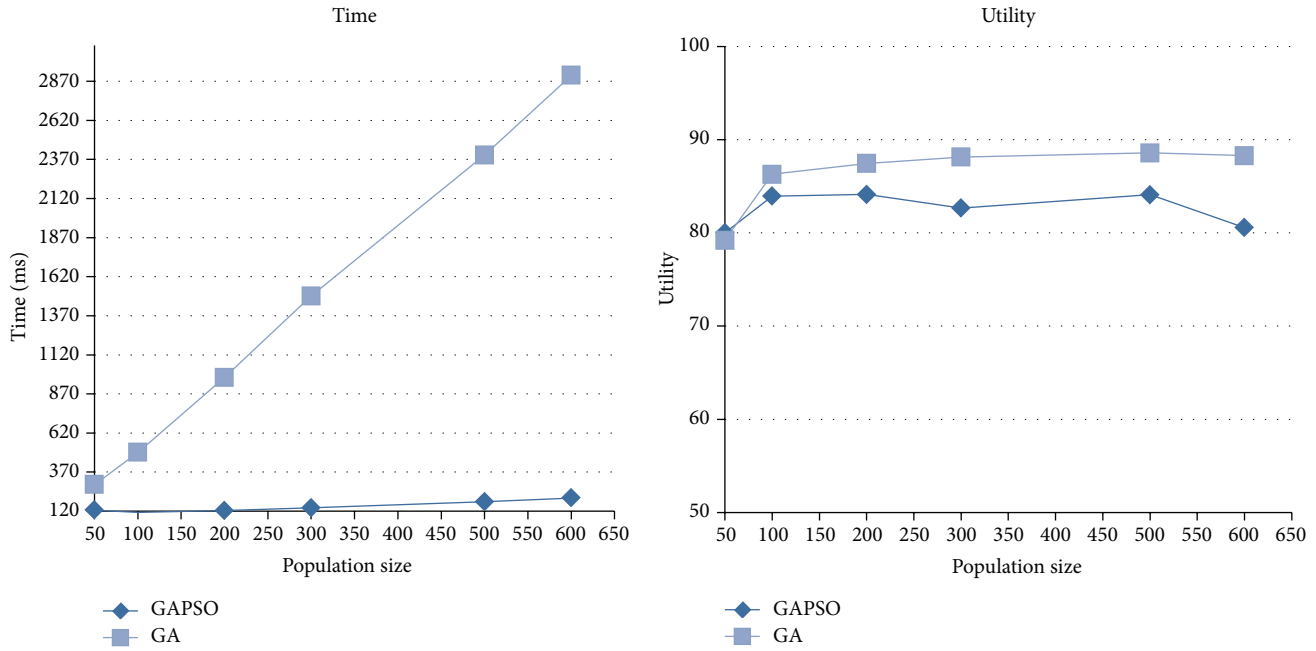

Figure 7: Comparison between GA and GAPSO on effect of population size.

cycle of the global best. On the other hand, in the second experiment (see Figure 4), increasing the population size will most likely increase the execution time; AUC-GAPSO took more time to apply the GA operators and calculate the PSO variables of each chromosome in the population. Figure 5 shows that increasing in the population or generation will not guarantee an improvement of the average fitness, and it could even make it worse.

5.1. AUC-GA and AUC-GAPSO Comparison. For comparing AUC-GAPSO to only genetic based solution AUC-GA, we conducted two experiments for AUC-GA and two experiments for AUC-GAPSO. First experiment set the population to 200 and the generation of $[50 \cdots 600]$ while the second experiment set the generation to 200 and the population of $[50 \cdots 600]$. Figure 6 shows the result of the first experiment, in both AUC-GA and AUC-GAPSO, related to increasing the generations. Figure 7 shows the result of the second experiment in both AUC-GAPSO and AUC-GA, related to increasing the population size. By comparing the graphs in Figures 6 and 7, we can observe that AUC-GAPSO has better performance than AUC-GA, especially in the time spent to extract the fittest matches. Therefore, we can conclude that it is better to use AUC-GAPSO rather than AUC-GA.

We can also see from Figures 6 and 7 that increasing the population and generation will not guarantee an improvement of the fitness; sometimes it even makes it worse. Moreover, even if AUC-GAPSO reduces the time, it is important to choose the right generation and population size, which means that we have to be careful in choosing the generation and population size. We think that it is better to use AUC-GAPSO with high population size and moderate generation to extract winners. This trade-off decision will prospectively guarantee good fitness in short time.

6. Conclusion

In this paper, we proposed a hybrid evolutionary algorithm AUC-GAPSO, which combined genetic and particle swarm optimization, to solve the winner determination problem in multiunit double auction. The AUC-GAPSO algorithm used the property of evolutionary algorithms that matches bidders in a short time. The proposed solution is focusing on maximizing all traders' utility or fitness through optimization based on hybridization of GA operations and PSO parameters. The algorithmic components, including real-coded representation, selection, crossover, mutation operators, and particles variables, all contribute to achieving a runtime improvement. Experimental results demonstrated that AUC-GAPSO in double auction incorporates high performance concerning the time of execution. In addition, AUC-GAPSO reduced the time when compared to AUC-GA. Moreover, we exposed how it is important to be careful in choosing the right generation and population size. In the future, one can solve multi-item (combinatorial) multiunit double auction which can be achieved by introducing multiobjective evolutionary algorithm.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work was supported by Deanship of Scientific Research and Research Center of College of Computer and Information Sciences, King Saud University. The authors are grateful for this support.

References

[1] P. R. Wurman, "Online auction site management," in *The Internet Encyclopedia*, B. Hossein, Ed., John Wiley & Sons, New York, NY, USA, 2004.

[2] J. H. Choi, H. Ahn, and I. Han, "Utility-based double auction mechanism using genetic algorithms," *Expert Systems with Applications*, vol. 34, no. 1, pp. 150–158, 2008.

[3] M. A. Rigi, S. Mohammadi, and M. Delgir, "Designing fuzzy utility-based double auctions using particle swarm optimization algorithm," in *Proceedings of the International Conference on Innovations in Information Technology (IIT '09)*, pp. 110–114, Al Ain, United Arab Emirates, December 2009.

[4] S. Gjerstad and J. Dickhaut, "Price formation in double auctions," *Games and Economic Behavior*, vol. 22, no. 1, pp. 1–29, 1998.

[5] J. E. Teich, H. Wallenius, J. Wallenius, and O. Koppius, "Emerging multiple issue E-auctions," Erasmus Research Institute of Management Report, 2003.

[6] M. Ykhlef and R. Alqifari, "Genetic algorithm based optimization for E-auction," in *Proceedings of the IADIS International Conference on E-Commerce*, Rome, Italy, 2011.

[7] D. Boughaci, L. Slaouti, and K. Achour, "A hybrid parallel genetic algorithm for the winner determination problem in combinatorial auctions," in *Proceedings of the 31st SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence (AI '11)*, pp. 65–78, Springer, December 2011.

[8] D. Boughaci, "Metaheuristic approaches for the winner determination problem in combinatorial auction," in *Artificial Intelligence, Evolutionary Computing and Metaheuristics: In the Footsteps of Alan Turing* , vol. 427, pp. 775–791, Springer, Berlin, Germany, 2013.

[9] M. Mitchell, *An Introduction to Genetic Algorithms*, MIT Press, Boston, Mass, USA, 1998.

[10] F. van den Bergh and A. P. Engelbrecht, "A cooperative approach to participle swam optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 225–239, 2004.

[11] A. Banks, J. Vincent, and C. Anyakoha, "A review of particle swarm optimization. Part II. Hybridisation, combinatorial, multicriteria and constrained optimization, and indicative applications," *Natural Computing*, vol. 7, no. 1, pp. 109–124, 2008.

[12] R. C. Eberhart and Y. Shi, "Comparison between genetic algorithms and particle swarm optimization," in *Proceedings of the 7th International Conference on Evolutionary Programming (EP '98)*, pp. 611–616, 1998.

[13] M. Settles and T. Soule, "A hybrid GA/PSO to evolve artificial recurrent neural networks," in *Intelligent Engineering Systems Through Aritificial Neural Networks*, vol. 3, pp. 51–56, 2003.

[14] R. Hassan, B. Cohanim, O. de Weck, and G. Venter, "A comparison of particle swarm optimization and the genetic algorithm," in *Proceedings of the Structural Dynamics and Materials Conference*, Austin, Tex, USA, 2005.

[15] M. Settles and T. Soule, "Breeding swarms: a GA/PSO hybrid," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '05)*, pp. 161–168, June 2005.

[16] X. Wang, M. Huang, Y. Ma, and Z. Tan, "A double auction method for resource management and bidding strategy on grid resources," in *Proceedings of the 4th International Conference on Genetic and Evolutionary Computing (ICGEC '10)*, pp. 422–425, December 2010.

[17] M. Bichler, J. Kalagnanam, H. Soo Lee, and J. Lee, "Winner determination algorithms for electronic auctions: a framework design," in *E-Commerce and Web Technologies: Third International Conference, EC-Web 2002 Aix-en-Provence, France, September 2–6, 2002 Proceedings*, vol. 2455, pp. 37–46, Springer, Berlin, Germany, 2002.

[18] P. Steve, S. Elizabeth, and P. Simon, "Using genetic programming to optimise pricing rules for a double auction market," in *Proceedings of the Workshop on Agents for Electronic Commerce*, Pittsburgh, Pa, USA, 2003.

[19] K. Jayant and D. C. Parkes, "Auctions, bidding and exchange design," in *Handbook of Quantitative Supply Chain Analysis: Modeling in the E-Business Era*, vol. 74 of *International Series*

in Operations Research & Management Science, pp. 143–212, Springer, Berlin, Germany, 2004.

[20] A. Movaghar and S. Khanpour, "Design and implementation of optimal winner determination problem in combinatorial e-auction," *World Academy of Science, Engineering and Technology*, vol. 20, pp. 57–60, 2006.

[21] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, vol. 4, pp. 1942–1948, IEEE, Perth, Australia, December 1995.

[22] J. E. Onwunalu and L. J. Durlofsky, "Application of a particle swarm optimization algorithm for determining optimum well location and type," *Computational Geosciences*, vol. 14, no. 1, pp. 183–198, 2010.

[23] C. Blum and X. Li, *Swarm Intelligence in Optimization*, Natural Computing Series, Springer, Berlin, Germany, 2008.

[24] T. Zhang and B. W. Brorsen, "Particle swarm optimization algorithm for agent-based artificial markets," *Computational Economics*, vol. 34, no. 4, pp. 399–417, 2009.

[25] M. Sudhakaran, P. Ajay-D-Vimalraj, and T. G. Palanivelu, "GA and PSO culled hybrid technique for economic dispatch problem with prohibited operating zones," *Journal of Zhejiang University: Science A*, vol. 8, no. 6, pp. 896–903, 2007.

[26] T. Buer and G. Pankratz, "Solving a bi-objective winner determination problem in a transportation procurement auction," *Logistics Research*, vol. 2, no. 2, pp. 65–78, 2010.

[27] D. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison Wesley, Reading, Mass, USA, 1989.

[28] P. Umapathy, C. Venkataseshaiah, and M. Arumugam, "Particle swarm optimization with various inertia weight variants for optimal power flow solution," *Discrete Dynamics in Nature and Society*, vol. 2010, Article ID 462145, 15 pages, 2010.

[29] K. Premalatha and A. M. Natarajan, "Hybrid PSO and GA for global maximization," *International Journal of Open Problems in Computer Science and Mathematics*, vol. 2, no. 4, pp. 597–608, 2009.

[30] F. Kui and N. Guihua, "Application of particle swarm optimization algorithm in e-commerce negotiation," in *Proceedings of the 6th Wuhan International Conference on E-Business*, pp. 225–231, 2007.

[31] R. C. Eberhart and Y. Shi, "Comparing inertia weights and constriction factors in particle swarm optimization," in *Proceedings of the IEEE International Conference Evolutionary Computation (CEC '00)*, pp. 84–88, La Jolla, Calif, USA, July 2000.

[32] Y. Shi and R. C. Eberhart, "Parameter selection in particle swarm optimization," in *Evolutionary Programming VII*, V. Porto, N. Saravanan, D. Waagen, and A. E. Eiben, Eds., vol. 1447 of *Lecture Notes in Computer Science*, pp. 591–600, Springer, Berlin, Germany, 1998.

[33] K. Premalatha and A. M. Natarajan, "Hybrid PSO and GA models for document clustering," *International Journal of Advances in Soft Computing and its Applications*, vol. 2, no. 3, pp. 302–320, 2010.

[34] P. Huang, A. Scheller-Wolf, and K. Sycara, "Design of a multi-unit double auction e-market," *Computational Intelligence*, vol. 18, no. 4, pp. 596–617, 2002.