Hindawi Publishing Corporation
Journal of Industrial Engineering
Volume 2014, Article ID 128542, 12 pages
http://dx.doi.org/10.1155/2014/128542

Research Article

A Multiobjective Iterated Greedy Algorithm for Truck Scheduling in Cross-Dock Problems

B. Naderi,[1] Shadi Rahmani,[2] and Shabnam Rahmani[2]

[1] *Department of Industrial Engineering, Faculty of Engineering, University of Kharazmi, Karaj, Iran*
[2] *Department of Industrial Engineering, Islamic Azad University, South Tehran Branch, Tehran, Iran*

Correspondence should be addressed to B. Naderi; bahman_naderi62@yahoo.com

Received 27 January 2014; Accepted 8 April 2014; Published 8 May 2014

Academic Editor: Wen-Chiung Lee

The cross-docking system is a new distribution strategy which can reduce inventories, lead times, and improve responding time to customers. This paper considers biobjective problem of truck scheduling in cross-docking systems with temporary storage. The objectives are minimizing both makespan and total tardiness. For this problem, it proposes a multiobjective iterated greedy algorithm employing advance features such as modified crowding selection, restart phase, and local search. To evaluate the proposed algorithm for performance, it is compared with two available algorithms, subpopulation particle swarm optimization-II and strength Pareto evolutionary algorithm-II. The comparison shows that the proposed multiobjective iterated greedy algorithm shows high performance and outperforms the other two algorithms.

1. Introduction

Efficiently managing the flow of products is one of the most essentialsteps in supply chain management. How this flow is handled is basically affected by transportation networks and distribution structures. Hence, any action contributing to the improvement of these structures such as the execution of cross-docking systems is considered worthwhile. In cross-docking system, products are received by inbound trucks in the receiving dock; then, they are unloaded, sorted, and reorganized based on customer demands. Afterwards, these products are loaded into the outbound trucks for delivery to customers, without being actually held as inventory at warehouse. If any item is held in storage, it takes usually a short time, generally less than 24 h.

In comparison with traditional warehousing strategy, cross-docking systems can cut down or remove storing and retrieving functions, the two most expensive warehousing operations, by synchronizing the flows of inbound and outbound trucks. As a result, not only total operational costs are decreased as a result of reduction of a considerable level of inventory in the distribution system, but also the customers can be served by more precise and on-time shipment deliveries. The cross-docking system is the best way to handle high volume of items in a short time, reduce cost and space required for inventory (or eliminate storage), increase throughput, and improve efficiency by increasing level of customer satisfaction [1]. Thus, cross-docking becomes an attractive alternative to warehousing.

The problem of truck scheduling in the cross-docking systems is to determine the sequence of inbound/outbound trucks to unload/load their products. Besides, the assignment of product transshipment is determined as well. It is also assumed that there is temporary storage in front of the shipping dock. If a product arriving at the shipping dock is not intended for loading into the outbound truck currently at the dock, the product is stored in the temporary storage until the appropriate outbound truck comes into the shipping dock. The truck docking pattern employed requires that both inbound and outbound trucks must stay in docks until they finish their task once they come into docks. According to Yu and Egbelu [1], the general framework of this cross-docking system is depicted in Figure 1.

Most of the papers in the problem of scheduling trucks in a cross-docking system consider a single optimization criterion, although in practice the decision maker often faces several (usually conflicting) criteria. Therefore, this paper

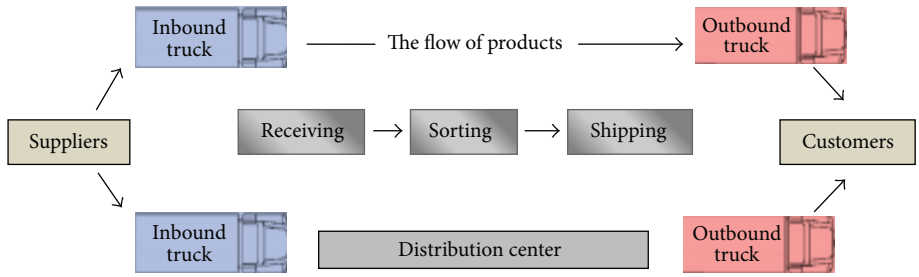

Figure 1: Typical flows in a cross-docking system.

addresses a biobjective problem of truck scheduling in a cross-docking system. The first objective is to minimize makespan (i.e., the length of time from unloading the first product from the first inbound truck until loading the last product on the last outbound truck). The second objective is to minimize the total tardiness of outbound trucks. The tardiness of a truck is equal to the difference between departure time of the truck and its due date.

To solve the problem under consideration, we propose a novel multiobjective algorithm based on the iterated greedy algorithm. An efficient management of the Pareto front, a modified crowding selection operator, an effective local search, and other techniques are applied in order to attain high quality and well spread Pareto fronts. The performance of this algorithm is compared with the subpopulation particle swarm optimization-II (SPPSO-II) proposed by Boloori Arabani et al. [2] and strength Pareto evolutionary algorithm-II (SPEA-II) proposed by Boloori Arabani et al. [3]. In order to evaluate the performance of these algorithms and make an analytical comparison, two advance performance measures are used (hypervolume indicator and epsilon indicator).

The rest of the paper is organized as follows. Section 2 reviews the literature of truck scheduling in cross-docking system. Section 3 proposes the multiobjective iterated greedy algorithm. Section 4 conducts some experiments to calibrate and evaluate the algorithms. Finally, Section 5 contains conclusions and future trends.

2. Literature Review

With the advent of cross-docking concept, many researchers have placed their emphasis on this field. After reviewing the literature, only few papers studied the problem of multiobjective truck scheduling in cross-docking systems. In the case of cross-docking systems, Yu [4] defined 32 different cross-docking models based on the number of docks available at the distribution facility, the dock holding pattern for trucks, and the existence of temporary storage. He focused on three models and applied several solution approaches to solve them. The objective of the study is to find the best truck docking sequence for both inbound and outbound trucks to minimize total operation time or to maximize the throughput of the cross-docking system.

Yu and Egbelu [1] studied a cross-docking system that has only one receiving dock and only one shipping dock and also temporary storage in front of the shipping dock. The objective was to find the best inbound and outbound trucks sequence to minimize makespan. The product assignments from inbound trucks to outbound trucks are also determined simultaneously with the docking sequences of the inbound and outbound trucks. To solve this problem, they developed three different solution approaches including mathematical model, the complete enumeration method, and a heuristic algorithm. This problem was then studied by Boloori Arabani et al. [5]. They developed five metaheuristic algorithms to solve the problem and made a comparison with the heuristic algorithm proposed by Yu and Egbelu [1]. Vahdani and Zandieh [6] develop five other metaheuristics for the same problem and compared them with the primary heuristic algorithm proposed by Yu and Egbelu [1].

There are some papers considering multiobjective cross-docking systems. Boloori Arabani et al. [7] consider a multiobjective cross-docking scheduling problem with a just-in-time approach. Earliness and tardiness were considered as the two objectives. They developed three metaheuristics, genetic algorithm, particle swarm optimization, and differential evolution algorithm. They evaluated these algorithms by applying the Taguchi plan and ANOVA tables and finally concluded that the particle swarm optimization outperformed the other two algorithms.

Boloori Arabani et al. [2] considered cross-docking systems where the objectives were minimization of the makespan and the total lateness. They proposed three multiobjective algorithms including subpopulation genetic algorithm-II (SPGA-II), subpopulation particle swarm optimization-II (SPPSO-II), and subpopulation differential evolution algorithm-II (SPDE-II). To evaluate the performance of these multiobjective algorithms, they used four measures to compare the algorithms. The results demonstrated that SPPSO-II outperformed the others. More recently, Boloori Arabani et al. [3] considered the same problem again. This time, they proposed three other multiobjective algorithms including nondominated sorting genetic algorithm-II (NSGA-II), strength Pareto evolutionary algorithm-II (SPEA-II), and subpopulation genetic algorithm-II (SPGA-II) to solve the problem. The results showed that SPEA-II worked better than the others.

As we reviewed the literature of multiobjective cross-docking systems, two recent multiobjective algorithms of SPPSO II and SPEA II proposed by Boloori et al. [2, 3] are state-of-the art algorithms.

3. Multiobjective Iterated Greedy Algorithm

This section proposes a multiobjective algorithm to solve the problem under consideration. This algorithm is an effective metaheuristic in form of iterated greedy algorithm (IGA). This algorithm has demonstrated to be very efficient for several combinatorial problems, including the flowshop scheduling problem with makespan objective (Ruiz and Stützle [8]). Framinan and Leisten [9] and Minella et al. [10] implemented IGA to solve a multiobjective flowshop scheduling problem.

IGA starts from a single solution as its working solution and consists of iteratively removing some elements of the working solution randomly (called destruction phase) and then adding so that a new complete and hopefully better solution is generated (called reconstruction phase). Finally, this new solution probabilistically replaced the working solution even if it is worse than the working solution. This procedure iterates until a stopping criterion, commonly a time limit, is met.

In a single-objective case, the quality of a solution is directly obtained by the objective function value. It can easily concluded that a solution x_1 is better than a solution x_2 if $f(x_1) < f(x_2)$ and the optimal solution is one that has the best objective function value. However, in multiobjective cases, these conclusions become more complicated. It should be in such a way that all the objectives are considered simultaneously [11]. To properly compare two solutions, some definitions are needed. A solution x_1 is said to dominate a solution x_2 if solution x_1 is not worse than x_2 for all the objectives and solution x_1 is better than x_2 for at least one objective. Moreover, in comparison, a new definition is faced. Two solutions x_1 and x_2 are said to be incomparable if x_1 is better than x_2 in some objectives while x_2 is better than x_1 in the other objectives. Thus, in multiobjective problems, the search is to find a set of solutions (called Pareto optimal set) that includes all nondominated solutions of solution space.

The main idea behind the proposed multiobjective IGA is handling a population of nondominated solutions as a working set instead of just a single solution. At each iteration, one solution from the working set is selected for further processing. The selection is done by a mechanism that accelerates the search and to maximize the spread of the final Pareto set. The selected solution has then undergone the greedy phase, in which some elements are eliminated. Next, the reconstruction procedure creates a whole set of nondominated solutions by inserting each removed element into a population of partial solutions. The working set is updated with the recently generated set of nondominated solutions from the reconstruction procedure. After the greedy phase, the working set is updated, possibly with new solutions. Thus, the solution selected previously for the greedy phase might not be exist in the working set anymore because of being dominated by other new solutions. As a consequence, the selection operator is employed again to select one solution that will go through local search. These two phases, namely, greedy and local search, are repeated until a termination criterion is satisfied. Algorithm 1 shows the outline of the proposed IGA.

Procedure: the multi-objective iterated greedy algorithm

Initialize the working set
While the stopping criterion is not met **do**
    Select a solution from the working set
    Deconstruct and Reconstruct the solution
    Update the working set
    Select a solution from the working set
    Go through local search phase
    Update the working set
Endwhile

Algorithm 1: The outline of the proposed IGA.

3.1. Initialization. Before describing how to generate an initial solution, it is necessary to show the representation scheme of solutions in this algorithm. As mentioned earlier, there are two decisions in the problem under consideration, the sequence of inbound and outbound trucks as well as the assignment of product transshipment. Our strategy to make the decisions is to determine the sequences by the algorithm and the assignment by a rule.

Thus, an encoded solution consists of two parts such that the first part is related to the inbound trucks and the second part is related to the outbound trucks. Figure 2 shows the encoding scheme for a problem with R inbound trucks and S outbound trucks. For this problem, there are $R! * S!$ different sequences.

To determine the assignment of product transshipment, products unloaded from early inbound trucks are transshipped to the first outbound truck available to load its products. To describe how to decode an encoded solution (i.e., calculation of the two objective functions), see Notations.

The departure time of the trucks determines makespan and total tardiness. The departure time of inbound trucks is calculated as such. Consider

$$C_{\text{in}[1]} = \sum_{k=1}^{N} r_{[1]k} u_k, \tag{1}$$

$$C_{\text{in}[i]} = C_{\text{in}[i-1]} + D + \sum_{k=1}^{N} r_{[i]k} u_k, \quad (2 \le i \le R). \tag{2}$$

The departure times of the first inbound truck in (1) and subsequent inbound trucks are determined by (2). To calculate the departure time of outbound trucks, we have

$$\begin{aligned} C_{\text{out}[j]} &= \max(h_1, h_2), \\ h_1 &= \max_{1 \le i \le R} \left\{ v_{[i][j]} \left(C_{\text{in}[i]} - \sum_{k=1}^{N} r_{[i]k} u_k + \sum_{k=1}^{N} t_{[i][j]k} l_k + V \right) \right\}, \\ h_2 &= C_{\text{out}[j-1]} + D + \sum_{k=1}^{N} s_{[j]k} l_k. \end{aligned} \tag{3}$$

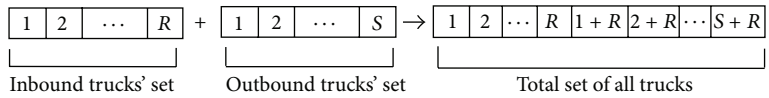

Figure 2: The representation scheme.

The departure time of jth outbound truck depends on the departure time of both preceding outbound trucks and inbound trucks that deliver products to jth outbound truck. Finally, makespan becomes the departure time of the last outbound truck.

$$C_{\max} = C_{\text{out}[S]}. \tag{4}$$

Also, the total tardiness becomes the summation of each outbound truck's tardiness. Consider

$$TT = \sum_{j=1}^{S} \left(C_{\text{out}[j]} - d_{[j]}\right). \tag{5}$$

The due dates of outbound trucks are generated by the following formula:

$$d_{[j]} = \frac{\left(\left(\sum_{k=1}^{N} s_{[j]k} l_k / N\right) * V\right) + D}{\text{random}}, \tag{6}$$

where "random" is a random number from a uniform distribution over range (2, 3). The initial solution is generated at random. That is, two random permutations of inbound and outbound trucks are considered.

3.2. Selection Mechanism. To select a quality Pareto set, it is necessary to consider both objective function and spread of solutions in the set. Hence, the fitness of each solution is defined according to these two factors. To determine the fitness of each solution in the set, we use modified crowding distance assignment (MCDA) procedure. This method is an extension of the well-known crowding distance operator initially presented by Deb et al. [12]. Based on the fitness, a solution is selected for the greedy phase. A solution with greater fitness has more chance of being selected.

The original method divides the working set into dominance levels; that is, the set of nondominated solutions forms the first-level Pareto front. Once we eliminate these elements, we have another nondominated set of solutions, which correspond with the second-level Pareto front. This procedure is repeated until all solutions are assigned to a Pareto front. Subsequently, the crowding distance operator assigns a value to each solution of the working set according to the distance between it and its nearest neighbors belonging to the same Pareto front level. Such method favors the selection of the most solitary solutions of the first frontier. The idea is that isolated solutions need to be further explored in order to close gaps in the objective solution space. The main drawback of this technique is that it does not keep track of how many times a solution has been previously selected; because of that, it might keep selecting it again and again. After selection, IGA employs the greedy and local search phases. Thus, applying the standard crowding distance procedure results in an algorithm that gets easily stuck, as if no improvements are found after the greedy and local search phases, the Pareto fronts do not change, and the same solution is selected repeatedly.

To avoid this, Minella et al. [10] added a selection counter to each solution which counts the number of times each solution has been selected. In this way, the probability of selecting a solution from the working set decreases as the selection counter increases. The proposed MCDA is manifested in pseudocode form in Algorithm 2.

3.3. Greedy Phase. The greedy phase works in two steps: first, a block of d elements is randomly eliminated from the MCDA-selected solution. It should be note that $d/2$ elements are chosen from inbound truck sequence and $d/2$ elements are chosen from outbound truck sequence for removal. The second step iteratively reconstructs the solution by reinserting, one by one, all the d removed elements into all possible positions of a group of partial solutions (because our proposed method actually maintains a population of partial solutions to which each removed element is reinserted). Also, it should be noted that the elements which belong to inbound/outbound truck sequence must be reinserted into all possible positions of inbound/outbound truck sequence. This process is repeated until the last removed element is inserted and a set of complete solutions is obtained.

To overcome the size of the partial solutions that grows exponentially, each time a set of partial solutions is created, only the nondominated partial solutions are kept and the dominated ones are discarded. In the next step, the next removed element is only reinserted in the nondominated partial solutions. Algorithm 3 explains the pseudocode of this greedy phase.

Afterwards, the solution set acquired from the greedy phase is appended to the working population and the dominated elements are eliminated. Finally, the MCDA is employed on this working set and a new solution is selected for the local search phase.

3.4. Local Search Phase. A simple local search phase is proposed to improve the search in the space close to the selected solution. The local search phase consists in randomly selecting each time, one element from inbound trucks sequence of the selected solution, and reinserting it into the n_{neigh} adjacent positions to the right and to the left of the original position in the related sequence, where n_{neigh} is a user-specified parameter. Then this operation is applied for outbound trucks sequence. The above procedure is repeated Selection Counter times.

Procedure: Modified Crowding Distance Assignment

Working Set := *Current set of solutions*
For each $m \in$ *objectives*
Sort Working Set using the current objective m
Set the distance of the extreme elements on the Working Set to ∞
$f_m^{\max}$:= *maximum value for objective* m *in Working Set*
$f_m^{\min}$:= *minimum value for objective* m *in Working Set*
// *Calculate the distance of all other elements in the Working Set*
For $i := 2, \ldots,$ *Working Set*$_{LastElement-1}$

$$Working\ Set_i \cdot distance := Working\ Set_i \cdot distance + \frac{(Working\ Set_{i+1} \cdot Objective_m - Working\ Set_{i-1} \cdot Objective_m)}{(f_m^{\max} - f_m^{\min})}$$

Endfor
Endfor
// *Adjust the fitness value with the selection counter in each individual*
// *First obtain the extreme distance values(for normalizing)*
Max *Dist* := max*(Working Set* · *distance)*
Min *Dist* := min*(Working Set* · *distance)*
For each $i \in$ *Working Set*
// *The infinit values of the extreme solutions should be replaced*
// *by the maximum acceptable value*
If *Working Set*$_i$· *distance* = ∞ *then*

$$Working\ Set_i \cdot fitness := \frac{1}{(Working\ Set_i \cdot SelectionCounter + 1)}$$

Else

$$Working\ Set_i \cdot fitness := \frac{(Working\ Set_i \cdot distance + \text{Min } Dist)/(\text{Max } Dist + \text{Min } Dist)}{(Working\ Set_i \cdot SelectionCounter + 1)}$$

Endif
Endfor

Algorithm 2: Modified crowding distance assignment procedure.

Procedure: *Greedy Phase*

Selected Solution := *MCDA-selected solution*
Partial Solution := *destruct Selected Solution by removing a block of* d *consecutive elements*
Destructed Element := *Set of d elements removed from Selected Solution*
Partial Solution Set := *Partial Solution* // *Set of partial solutions used in this phase*
For each $i \in$ *DestructedElements*
For each $n \in$ *Partial Solution Set*
Counter := *0*
For each *position* j *of Partial Solution Set*$_n$
New Partial Solution Set$_{Counter}$:= *Insert element* i *in position* j *of Partial Solution Set*$_n$
Counter := *Counter* + *1*
Endfor
Endfor
Partial Solution Set := *remove dominated partial solutions of New Partial Solution Set*
Endfor

Algorithm 3: Greedy phase pseudocode.

Similar to the greedy phase, instead of keeping one full solution in this local search phase, we keep a local working set of solutions. At each step, a removed element is reinserted, and we add the new solutions to the local working set. For each of the removed elements, $n_{\text{neigh}} * 2$ new solutions are added to the local working set. After the local search phase, dominated elements are removed from the local working set and the remaining solutions are finally added to the algorithm's working set.

3.5. Restart Phase. The last phase is the restart procedure and consists in archiving the current working set and then

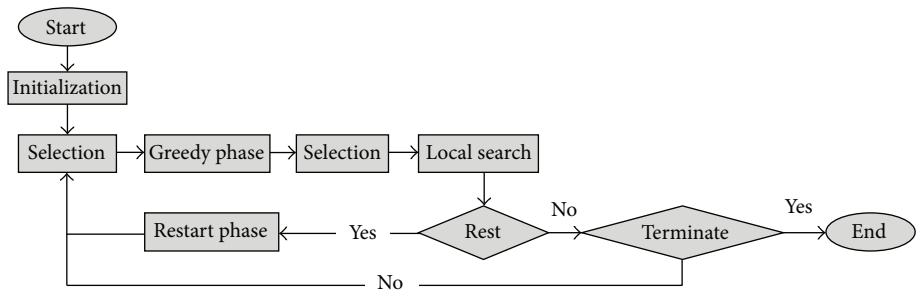

Figure 3: IGA with additional restart phase.

Table 1: Parameters and their level of IGA algorithm.

Parameters	Level's number	Levels of parameters
d	1	4
	2	6
	3	8
	4	10
	5	12
n_{neigh}	1	1
	2	2
	3	3
	4	4
	5	5

creating new one with randomly generated solutions. This is the simplest possible restart procedure that still allows the algorithm to escape from a situation in which the current working set is stalled. One of the difficulties of this phase is to understand when to implement the restart. A very simple method is to restart when the working population has not changed for a given number of iterations. The issue of determining when a working set has not changed is not trivial, as this can be calculated in a number of ways. Based on what proposed in Minella et al. [10], we choose to trigger the restart when the size of the working set has not changed for a number of user-specified iterations. Hence, initial tests demonstrate that this number of iterations can be set based on the size of the input instance to $(R + S + N) * 3$. Figure 3 represents the complete flowchart of the final version of the IGA algorithm with all of its phases.

4. Computational Evaluation

This section assesses the performance of the proposed IGA. To this purpose, we bring the SPPSOII and SPEAII proposed by Boloori Arabani et al. [2, 3] for a relevant problem into the experiment. The algorithms are implemented in Microsoft visual studio C++ 2010 and run on a PC with 3.0 GHz Dual Core CPU and 3 GB of random access memory (RAM). For all the algorithms, we use the same stopping criterion which is a time limit equal to $3 * (R + S) * N$ seconds that depends on the size of the input problem set. By using those stopping criteria, we give more computation time for bigger problem sets which have larger solution spaces.

To compare the approximation solution methods (ASM), we require some performance measures (PM). They should be designed to assess the performance of tested ASM with no bias or misleading results. In multi-objective cases, PMs are challenging since each ASM gives an approximation Pareto set and these sets need to be compared. For example, consider two ASMs that provide two approximation Pareto sets A and B. If set A is dominated by set B, it can be easily concluded the ASM giving set B outperforms the one giving set A. But very often we encounter two incomparable approximation Pareto sets A and B; as a result, no clear conclusion could be seen to decide which of tested ASMs is more preferable. Zitzler et al. [13] carried out a comprehensive study of such existing performance measures. The results demonstrate that some metrics frequently used in multiobjective research are not Pareto-compliant; that is, in some cases, a non-Pareto-compliant performance measure can allocate a better value to a Pareto set B with respect to set A even if A dominates B. Again, this result highlights the fact that comparing Pareto sets is not an easy task at all.

After all, three PMs are known to give reliable analyses (Knowles et al. [11]). These three PMs, sorted from easy to complicated ones, are the following.

(1) *Dominance Ranking*. To compare two approximation Pareto sets A and B, one can rank a given set A over another set B by counting the number of points in B that is dominated by or equal to points in A.

(2) *Quality Indicator*. This is a function that assigns a real number to a full Pareto approximation set.

(3) *Empirical attainment Function*. The relative frequency that each region is attained by the approximation set is calculated.

In this paper, we employ two major types of quality indicators that were shown to be Pareto-compliant in Zitzler et al. [13]. The combination of quality indicators could provide more precise conclusion than using a single quality indicator alone. If the results of two quality indicators are in conflict with one another on preference ranking of two approximation sets, then it specifies that the two sets are incomparable. Before describing used quality indicators, two following definitions are presented.

Table 2: The size characteristics of five problem sets.

Problem set	Problem size			Total number of products
	Number of inbound trucks	Number of outbound trucks	Number of product types	
1	8	11	12	5293
2	15	15	15	5326
3	19	16	18	7964
4	17	16	19	10852
5	20	20	12	5314

Table 3: The average of hypervolume indicator that was obtained in each level of parameters.

Problem set	D					n_{neigh}				
	4	6	8	10	12	1	2	3	4	5
1	0.788	0.826	0.781	0.746	0.785	0.749	0.771	0.824	0.788	0.772
2	0.843	0.844	0.823	0.878	0.778	0.701	0.902	0.874	0.856	0.831
3	0.690	0.690	0.693	0.655	0.679	0.647	0.673	0.717	0.714	0.665
4	0.814	0.825	0.794	0.713	0.705	0.751	0.747	0.832	0.799	0.722
5	0.706	0.739	0.741	0.660	0.674	0.737	0.685	0.766	0.699	0.632
Average	0.768	0.785	0.766	0.732	0.724	0.717	0.755	0.803	0.771	0.724

Definition 1 (reference Pareto set). For a given instance, the reference Pareto set has been constructed from all nondominated solutions found from all tested methods and experiments.

Definition 2 (reference point). For a given instance, the best (worst) reference point is the best (worst) values available in the approximation Pareto sets for each objective function.

The first quality indicator is the hypervolume indicator (I_H) presented in Zitzler and Thiele [14] and, more precisely, its unary version from Zitzler et al. [15]. This indicator calculates the hypervolume (or area in biobjective cases) of the objective space dominated by a given Pareto set of points. Notice that in the comparison of two Pareto sets, a higher value of I_H indicates a better set. The hypervolume requires a reference point for closing the volume. In our case, this reference point is attained by multiplying the worst objective values by 20%. As the objective values are normalized, the maximum I_H value can be obtained by the product of the reference point values: $1.2 * 1.2 = 1.44$.

The second quality indicator is the so-called Unary Epsilon Indicator ($I_{\varepsilon 1}$), proposed by Knowles et al. [11]. It measures the minimum distance between a given Pareto set and the optimal or reference Pareto set. It is formally calculated as follows. Suppose that, without loss of generality, a minimization problem with n objectives, P is the Pareto front or a reference set and S is an approximation to the Pareto front. Actually, $I_{\varepsilon 1} = I_\varepsilon(S, P)$ where $I_\varepsilon(S, P) = \inf_{\varepsilon \in R}\{\forall\, x^2 \in P\ \exists x^1 \in S :\ x^1 \preccurlyeq x^2\}$ and $x^1 \preccurlyeq x^2 \leftrightarrow \forall 1 \le i \le n : z_i^1 \le \varepsilon \cdot z_i^2$. A value close to 1 means that the given Pareto set is close to the reference set. Hence, a lower value of $I_{\varepsilon 1}$ indicates a better set.

4.1. Parameter Tuning. This section studies the influence of parameter's values on the performance of the IGA. We employ the design of experiments (DOE) technique for the experiment, where the parameters affecting the performance of the IGA are tested in a full factorial experimental design which is later analyzed by means of the analysis of variance (ANOVA) technique. The proposed multiobjective IGA has two parameters: (1) the size of the destruction block (d), which has been tested with five levels: 4, 6, 8, 10, and 12, and (2) the size of the local search neighborhood or n_{neigh}, which has been tested at five levels: 1, 2, 3, 4, and 5. These parameters and their levels are given in Table 1.

As a result, each parameter is measured at five levels. This algorithm is run using different combinations of parameters. Therefore, this gives a total of 25 algorithm configurations. Each configuration is tested with the 2 instances and 5 different problem sets (as shown in Table 2) and 2 replicates. A total number of 500 results are therefore obtained. The assumptions of the instances are as follows.

Instance 1. The loading and unloading time per product is assumed to be equal for all products and is assumed to be one time unit in duration for each product type ($l_k = u_k = 1$; $k = 1, \ldots, N$). The truck's changeover time is considered to be 75 time units ($D = 75$), and the transfer time of products from the receiving dock to shipping dock through a set of conveyors is 100 time units ($V = 100$).

Instance 2. The loading and unloading time per product is assumed to be unequal for each product type and is assumed to be a random number to be chosen from a uniform distribution over range (1, 10). ($l_k \neq u_k = U[1, 10]$; $k = 1, \ldots, N$). The time for each truck's changeover is considered to be 60 time units ($D = 60$), and the time consumed to

Table 4: The average of epsilon indicator that was obtained in each level of parameters.

Problem set	D					n_{neigh}				
	4	6	8	10	12	1	2	3	4	5
1	1.089	1.111	1.120	1.129	1.113	1.116	1.113	1.102	1.126	1.107
2	1.072	1.067	1.073	1.066	1.076	1.088	1.060	1.065	1.071	1.071
3	1.083	1.080	1.079	1.082	1.076	1.080	1.081	1.076	1.078	1.085
4	1.042	1.045	1.042	1.050	1.050	1.046	1.045	1.040	1.046	1.053
5	1.066	1.059	1.060	1.066	1.064	1.060	1.066	1.051	1.062	1.077
Average	1.071	1.072	1.075	1.079	1.076	1.078	1.073	1.067	1.076	1.078

Table 5: Analysis of variance (ANOVA) for a statistical significance test of the parameters on the hypervolume indicator values.

Factor	*DF*	*SS*	*MS*	*F*	*P* value
D	4	0.1625	0.0406	0.80	0.524
n_{neigh}	4	0.4758	0.1189	2.35	0.054
Interaction	16	1.1571	0.0723	1.43	0.125
Error	375	18.9766	0.0506		
Total	399	20.7720			

Table 6: Analysis of variance (ANOVA) for a statistical significance test of the parameters on the epsilon indicator values.

Factor	*DF*	*SS*	*MS*	*F*	*P* value
D	4	0.0048	0.0012	0.59	0.668
n_{neigh}	4	0.0067	0.0016	0.82	0.512
Interaction	16	0.0385	0.0024	1.18	0.280
Error	375	0.7656	0.0020		
Total	399	0.8158			

transfer each product via conveyor from receiving dock to shipping dock is 80 time units ($V = 80$).

The experiments are implemented and the obtained nondominated Pareto sets are transformed into hypervolume and epsilon indicators. The average of hypervolume indicator that was obtained in each level of mentioned parameters is shown in Table 3. As stated before, if a parameter's level has the biggest hypervolume indicator, it will become the most effective level of the parameter. The average of epsilon indicator that was obtained in each level of mentioned parameters is shown in Table 4. As stated before, if a parameter's level has the least epsilon indicator, it will become the most effective level of the parameter. Based on the obtained average of the hypervolume values in each level of these parameters, we can conclude that better robustness of IGA is achieved when the levels of the parameters are set as follows: $d = 6$, $n_{\text{neigh}} = 3$. Based on the obtained average of the epsilon values in each level of these parameters, we can conclude that better robustness of IGA is achieved when the factors are set as follows: $d = 4$, $n_{\text{neigh}} = 3$.

For statistical significance test of the parameters, the analysis of variance (ANOVA) is carried out. The response variables of the ANOVA experiment are the hypervolume (Table 5) and Epsilon indicators (Table 6). Considering hypervolume, the result indicates that the parameters d and n_{neigh} do not have the significant impact on the robustness of IGA. Considering Epsilon indicator, the result shows that the parameters d and n_{neigh} do not have the significant effect on the robustness of IGA.

After the calibration, we can see from the results, apart from some minor exceptions, the best size for the destruction block resulting in being $d = 6$ and the size of the local search neighborhood resulting in being $n_{\text{neigh}} = 3$.

4.2. Data Generation. After adjustment parameters, in order to implement IGA and compare the performance of this algorithm with the existent meta-heuristic algorithms from the literature (SPPSOII and SPEAII) that are proposed by Boloori Arabani et al. [2, 3], twenty problem sets were generated randomly, in which the number of inbound and outbound trucks and also the number of product types are chosen from the range (8, 20). The sizes for each problem set are presented in Table 7 and also five different instances are generated in which their assumptions are as follows.

Instance 1. The loading and unloading time per product is assumed to be equal for all products and is assumed to be one time unit in duration for each product type ($l_k = u_k = 1$; $k = 1, \ldots, N$). The truck's changeover time is considered to be 75 time units ($D = 75$), and the transfer time of products from the receiving dock to shipping dock through a set of conveyors is 100 time units ($V = 100$).

Instance 2. The loading and unloading time per product is assumed to be equal for each product type and is assumed to be a random number to be chosen from a uniform distribution over range (1, 5). ($l_k = u_k = U[1, 5]$; $k = 1, \ldots, N$). The time for each truck's changeover is considered to be 90 time units ($D = 90$), and the time consumed to transfer each product via conveyor from receiving dock to shipping dock is 120 time units ($V = 120$).

Instance 3. The unloading and loading time per product is assumed to be equal for all product type and is assumed to be 3 and 5 time unit in duration for each product type, respectively ($u_k = 3; l_k = 5, k = 1, \ldots, N$). The truck's changeover time is considered to be 80 time units ($D = 80$), and the transfer time of products from the receiving dock to

Table 7: The size characteristics of twenty problem sets.

Problem set	Problem size			Total number of products
	Number of inbound trucks	Number of outbound trucks	Number of product types	
1	8	11	12	5293
2	9	14	8	5600
3	10	8	11	4923
4	11	10	10	3115
5	11	15	8	4786
6	12	9	11	6007
7	12	12	12	3060
8	13	11	13	2614
9	14	14	13	5040
10	15	15	15	5326
11	15	19	16	13374
12	16	17	16	5676
13	17	15	17	9944
14	17	16	19	10852
15	17	18	11	5377
16	18	16	14	5905
17	19	16	18	7964
18	19	19	13	6384
19	20	17	14	5488
20	20	20	12	5314

Table 8: The characteristics of five instances.

Parameters	Instances				
	Instance 1	Instance 2	Instance 3	Instance 4	Instance 5
u_k	1	$U[1, 5]$	3	$U[5, 10]$	$U[10, 15]$
l_k	1	$U[1, 5]$	5	$U[5, 10]$	$U[10, 15]$
D	75	90	80	60	110
V	100	120	115	85	150

shipping dock through a set of conveyors is 115 time units ($V = 115$).

Instance 4. The loading and unloading time per product is assumed to be unequal for each product type and is assumed to be a random number to be chosen from a uniform distribution over range (5, 10) ($l_k \neq u_k = U[5, 10]$; $k = 1, \ldots, N$). The time for each truck's changeover is considered to be 60 time units ($D = 60$), and the time consumed to transfer each product via conveyor from receiving dock to shipping dock is 85 time units ($V = 85$).

Instance 5. The loading and unloading time per product is assumed to be unequal for each product type and is assumed to be a random number to be chosen from a uniform distribution over range (10, 15) ($l_k \neq u_k = U[10, 15]$; $k = 1, \ldots, N$). The truck's changeover time is considered to be 110 time units ($D = 110$), and the transfer time of products from the receiving dock to shipping dock through a set of conveyors is 150 time units ($V = 150$). The characteristics of these instances are shown in Table 8.

4.3. Statistical Analysis of Algorithms. In this section, we compare our proposed algorithm with the best performing algorithms proposed by Boloori Arabani et al. [2, 3] (SPPSOII and SPEAII). The two algorithms are reimplemented in visual studio C++ 2010 following the original papers.

After running each of the applied multiobjective algorithms under their optimal parameters, we will have the final nondominated Pareto set of solutions for each problem set. As stated before, these Pareto sets must be evaluated, analyzed, and compared with each other by the means of hypervolume and epsilon indicators so that the effectiveness of each multiobjective algorithm can be clarified.

As mentioned before, each algorithm is run for 5 instances and 20 problem sets and the results are shown in Table 9 in order to analyze them. In this table, columns 2–4 present the average values of hypervolume indicator of all problem sets for IGA, SPPSOII, and SPEAII, respectively, and columns 5–7 present the average values of epsilon indicator of all problem set for IGA, SPPSOII, and SPEAII, respectively. Each value is averaged across 5 instances. As stated before, the higher the hypervolume indicator value and the lower the epsilon indicator value, the better the corresponding Pareto set.

Based on the obtained average of two quality indicators values in each algorithm, we can conclude that IGA favorably outperforms the other two algorithms with I_H of 1.425 and I_ε of 1. SPPSOII gains the second best results with I_H of 0.681

Table 9: The average values of hypervolume and the epsilon indicator of all problem sets for each algorithm.

Problem set	Performance indicators					
	I_H			I_ε		
	Algorithms					
	IGA	SPPSOII	SPEAII	IGA	SPPSOII	SPEAII
1	1.409	0.488	0.317	1	1.172	1.171
2	1.423	0.721	0.371	1	1.159	1.376
3	1.387	0.718	0.423	1	1.144	1.290
4	1.423	0.745	0.300	1	1.355	1.505
5	1.438	0.912	0.465	1	1.143	1.354
6	1.399	0.792	0.274	1	1.245	1.332
7	1.435	0.564	0.281	1	1.146	1.178
8	1.400	0.734	0.217	1	1.086	1.152
9	1.437	0.568	0.226	1	1.107	1.151
10	1.435	0.570	0.308	1	1.105	1.164
11	1.433	0.758	0.180	1	1.071	1.149
12	1.438	0.546	0.313	1	1.088	1.121
13	1.434	0.654	0.280	1	1.100	1.136
14	1.430	0.651	0.141	1	1.069	1.144
15	1.431	0.656	0.253	1	1.085	1.166
16	1.436	0.712	0.389	1	1.112	1.185
17	1.406	0.681	0.130	1	1.057	1.127
18	1.435	0.752	0.274	1	1.124	1.156
19	1.436	0.811	0.322	1	1.092	1.164
20	1.437	0.590	0.278	1	1.153	1.149
Average	1.425	0.681	0.287	1	1.131	1.208

Table 10: Analysis of variance (ANOVA) for a statistical significance test of the algorithms on the hypervolume indicator values.

Factor	*DF*	*SS*	*MS*	*F*	*P* value
Algorithms	2	13.35867	6.67933	1078.93	0.000
Error	57	0.35287	0.00619		
Total	59	13.71154			

Table 11: Analysis of variance (ANOVA) for a statistical significance test of the algorithms on the epsilon indicator values.

Factor	*DF*	*SS*	*MS*	*F*	*P* value
Algorithms	2	0.44602	0.22301	42.68	0.000
Error	57	0.29783	0.00523		
Total	59	0.74386			

and I_ε of 1.131. SPEAII provides the worst results among the tested algorithms with I_H of 0.287 and I_ε 1.208.

It can be noticed that according to the both quality indicators, IGA algorithm is the best performer, the second one in the ranking is SPPSO II while the worst is SPEAII. However, for small instances, we can see how the IGA performs much better. For the further analysis, we carry out the ANOVA. The related results show that there is statistically significant difference between the performances of the algorithms with *P*-value of zero. The results are provided in Tables 10 and 11.

Figure 4 shows the means and 95% confidence interval for the algorithms in terms of hypervolume and epsilon indicator. Notice that the overlapping of confidence intervals between any two algorithms indicates that there is no statistical difference between their performances. This figure indicates that there are strong and statistically significant differences between these three algorithms' performance, because the presented confidence intervals are not overlapping with each other in this figure. Clearly, the proposed IGA provides statistically better results than other methods considering both hypervolume and epsilon indicators.

As a conclusion and according to the results of the two analyzed measures, it can be stated that the IGA algorithm with average hypervolume and epsilon values of 1.425 and 1.00, respectively, can surmount the other two algorithms.

5. Conclusion and Future Trends

In this paper, we have proposed the iterated greedy algorithm for a biobjective scheduling problem in the cross-docking system with the objectives of minimizing the makespan and the tardiness in order to fill the current research gap in the case of multiobjective optimization problems. In the cross-docking system, inbound trucks are coming into the receiving dock while their products are unloaded. Then, the product items are categorized and sorted out in the temporary storage. Afterwards, they are loaded into the outbound trucks leaving the shipping dock. In fact, the scheduling problem is how to schedule and assign the inbound and outbound trucks in order to minimize the two objective functions. The effectiveness of the approach is established by comparing it with the best existing algorithms for the problem under consideration (SPPSOII and SPEAII) presented by Boloori Arabani et al. [2, 3]. For this purpose, 5 instances and 20 different problem sets were produced.

The performance of each of these multiobjective algorithms was analyzed and compared by means of two Pareto-compliant performance measures which demonstrate that the proposed IGA can relatively overwhelm other two algorithms.

Regarding future research trends in the case of cross-docking systems, some of the assumptions of this paper can be modified. For example, the arrival time of trucks can be changed to variable times. Moreover, instead of single receiving and shipping docks, multiple receiving and shipping docks can be implemented so that several inbound and outbound trucks can be dealt with concurrently. The capacity of temporary storage which in this paper is supposed to be infinite can be limited. Additionally, other conditions can be taken into account in which a particular set of inbound and outbound trucks commute between the specific set of cross-dock terminals and transport products in a distribution network. Furthermore, we can suggest more complicated and applicable versions of multiobjective algorithms in which hybrid metaheuristics are employed. Moreover, to obtain new areas of cross-docking systems, new constraints can be added to the problem such as time windows and maximum acceptable due dates for product delivery.

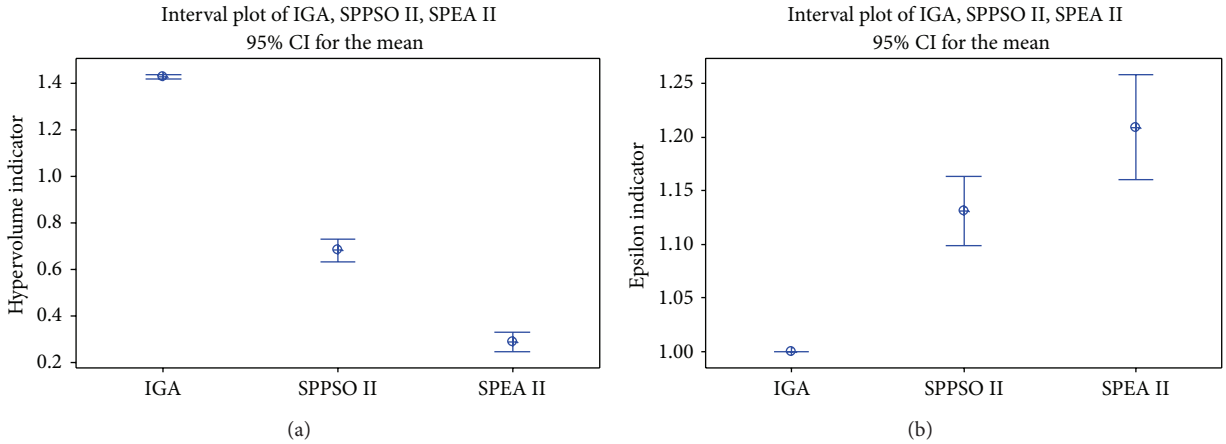

Figure 4: Means and 95% confidence interval for each algorithm in terms of the quality indicators.

Notations

$C_{\max}$: Makespan
TT: Total tardiness
R: The number of inbound trucks
S: The number of outbound trucks
N: The number of product types
D: Truck changeover time
V: Moving time of product from the receiving dock to the shipping dock per product unit
$[i]/[j]$: The ith/jth position in the sequence of inbound/outbound trucks
$d_{[j]}$: Due date for the jth positioned outbound truck in the outbound truck sequence
$r_{[i]k}$: The number of units of product type k that was initially loaded in the ith positioned inbound truck in the inbound truck sequence
$s_{[j]k}$: The number of units of product type k that was initially needed by the jth positioned outbound truck in the outbound truck sequence
$t_{[i][j]k}$: The total number of products type k that transfer from the ith positioned inbound truck in the inbound truck sequence to the jth positioned outbound truck in the outbound truck sequence
$v_{[i][j]}$: 1 if any product is transshipped for the ith positioned inbound truck and the jth positioned outbound truck, and 0 otherwise
l_k: The loading time per product unit type k
u_k: The unloading time per product unit type k
$C_{\text{in}[i]}$: The time at which the ith positioned inbound truck in the inbound truck sequence leaves the receiving dock
$C_{\text{out}[j]}$: The time at which the jth positioned outbound truck in the outbound truck sequence leaves the shipping dock.

Conflict of Interests

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

References

[1] W. Yu and P. J. Egbelu, "Scheduling of inbound and outbound trucks in cross docking systems with temporary storage," *European Journal of Operational Research*, vol. 184, no. 1, pp. 377–396, 2008.

[2] A. Boloori Arabani, M. Zandieh, and S. M. T. Fatemi Ghomi, "A cross-docking scheduling problem with sub-population multi-objective algorithms," *International Journal of Advanced Manufacturing Technology*, vol. 58, no. 5–8, pp. 741–761, 2012.

[3] A. Boloori Arabani, M. Zandieh, and S. M. T. Fatemi Ghomi, "Multi-objective genetic-based algorithms for a cross-docking scheduling problem," *Applied Soft Computing Journal*, vol. 11, no. 8, pp. 4954–4970, 2011.

[4] W. Yu, *Operational strategies for cross docking systems [Ph.D. dissertation]*, Iowa State University, 2002.

[5] A. R. Boloori Arabani, S. M. T. Fatemi Ghomi, and M. Zandieh, "Meta-heuristics implementation for scheduling of trucks in a cross-docking system with temporary storage," *Expert Systems with Applications*, vol. 38, no. 3, pp. 1964–1979, 2011.

[6] B. Vahdani and M. Zandieh, "Scheduling trucks in cross-docking systems: robust meta-heuristics," *Computers and Industrial Engineering*, vol. 58, no. 1, pp. 12–24, 2010.

[7] A. R. Boloori Arabani, S. M. T. Fatemi Ghomi, and M. Zandieh, "A multi-criteria cross-docking scheduling with just-in-time approach," *International Journal of Advanced Manufacturing Technology*, vol. 49, no. 5–8, pp. 741–756, 2010.

[8] R. Ruiz and T. Stützle, "A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem," *European Journal of Operational Research*, vol. 177, no. 3, pp. 2033–2049, 2007.

[9] J. M. Framinan and R. Leisten, "A multi-objective iterated greedy search for flowshop scheduling with makespan and flowtime criteria," *OR Spectrum*, vol. 30, no. 4, pp. 787–804, 2008.

[10] G. Minella, R. Ruiz, and M. Ciavotta, "Restarted Iterated Pareto Greedy algorithm for multi-objective flowshop scheduling

problems," *Computers and Operations Research*, vol. 38, no. 11, pp. 1521–1533, 2011.

[11] J. Knowles, L. Thiele, and E. Zitzler, "A tutorial on the performance assessment of stochastic multi-objective optimizers," Tech. Rep. 214, revised version, Computer Engineering and Networks Laboratory (TIK), ETH, Zurich, Switzerland, 2006.

[12] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[13] E. Zitzler, J. Knowles, and L. Thiele, "Quality assessment of Pareto set approximations," in *Multi-Objective Optimization: Interactive and Evolutionary Approaches*, pp. 373–404, Springer, Berlin, Germany, 2008.

[14] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 257–271, 1999.

[15] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. Da Fonseca, "Performance assessment of multiobjective optimizers: an analysis and review," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 2, pp. 117–132, 2003.