RAIRO-Oper. Res. 51 (2017) 833–856
DOI: 10.1051/ro/2016067

RAIRO Operations Research
www.rairo-ro.org

TRUCK ROUTING AND SCHEDULING FOR CROSS-DOCKING IN THE SUPPLY CHAIN: MODEL AND SOLUTION METHOD

Mehdi Yazdani[1], Bahman Naderi[2], Shabnam Rahmani[3] and Shadi Rahmani[3]

Abstract. Among various distribution networks, the idea behind the cross-docking for cost reduction is to decrease/eliminate inventory to the extent possible. In classical cross-dock, it is assumed that there is one truck for each supplier and customer. Yet, one truck for each supplier and customer can be very costly and consequently ineffective. Each truck likely can serve more than one supplier/customer in its pickup/delivery process. Therefore, to more actualize the cross-dock problem, it can be extended with the truck routing problem, *i.e.*, the truck scheduling in the cross-docking system and truck routing in the pickup/delivery process. Hence, this paper considers the integrated truck routing and scheduling problem. First, the problem is formulated as a mixed integer linear programming model. Using this model, we solve small-sized instances to optimality. Moreover, two metaheuristics, a reactive tabu search with path relinking and a generational genetic algorithm with a local search and restart phase, are proposed to solve large instances. The parameters of the proposed algorithms are tuned. Finally, the performance of the proposed algorithms is evaluated.

Mathematics Subject Classification. 90Bxx.

Received February 6, 2015. Accepted October 5, 2016.

1. Introduction

Nowadays, companies aim at decreasing costs by reducing inventory at every step of the operation, including distribution networks. The reason is that inventories can bring various costs to the supply chain system. On the other hand, in most manufacturing environments, direct product shipping from manufacturers to customers is difficult and costly since customers, such as retailers, may require ordering multiple trucks from different suppliers. This is not effective for both suppliers and customers. Thus, distribution centers are required to connect suppliers to customers. One such a distribution network in the supply chain system is the cross-dock.

In the cross-dock, inbound trucks transport products from suppliers to the cross-docking system and unload these products in the receiving dock. Then, the products are immediately sorted and categorized based on customer demands. Afterwards, the products are moved and loaded into outbound trucks departing from the shipping dock to deliver the products to customers [2, 9, 10]. Furthermore, it is assumed that there is a temporary storage in front of the shipping dock. If a product arriving at the shipping dock is not intended

Keywords. Supply chain, cross-docking, truck routing scheduling, metaheuristics.

[1] Department of Industrial Engineering, Faculty of Industrial and Mechanical Engineering, Qazvin Branch, Islamic Azad University, Qazvin, Iran. mehdi_yazdani2007@yahoo.com

[2] Department of Industrial Engineering, Faculty of Engineering, Kharazmi University, Tehran, Iran.

[3] Department of Industrial Engineering, South Tehran Branch, Islamic Azad University, Tehran, Iran.

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. Note that the products are usually stored for a short time in the cross-dock, generally less than 24 h [11, 24].

The cross-docking network is known to be the best way to handle high volume of items in a short time and reduce costs and space requirement for inventory with reducing inventory levels. It also improves efficiency by increasing level of customer satisfaction. However, in the classical cross-dock, it is assumed that there is one truck for each supplier and customer. Yet, assigning one truck for each supplier and customer increases the costs and consequently is ineffective. Each truck usually serves more than one supplier/customer in its pickup/delivery process. Therefore, to more actualize the cross-dock problem, it can be extended with truck routing problem, *i.e.*, the truck scheduling in the cross-docking system and truck routing in the pickup/delivery process.

In this paper, we consider the truck routing and scheduling problem. We first formulate the problem by a mixed integer linear programming (MILP) model. Using this model and specified software of CPLEX, we can solve small instances to optimality. Since this problem is known to be NP-hard, we then propose two effective metaheuristics to tackle large instances. The first metaheuristic is a genetic algorithm employing a novel crossover, mutation, advanced techniques like local search and generational scheme as well as a restarting phase. They keep an acceptable level of diversity in the population. The next metaheuristic is reactive tabu search algorithm equipped with path relinking and a new neighborhood search structure. To represent solutions of the classical cross-dock, the truck permutation is commonly used in the literature. This type of scheme cannot be used for the problem under consideration since truck routing is also added to the problem. We define a novel encoding scheme that can best encode solutions. After tuning the parameters of the proposed algorithms, their performance is evaluated over a set of small and large instances.

The rest of this paper is organized as follows. Section 2 provides a brief literature review of cross-docking problems. Section 3 presents the mathematical formulation of the problem. Section 4 proposes the two metaheuristics. Section 5 tunes parameters of the algorithms. Section 6 conducts the experiments. Finally, Section 7 concludes the paper and defines future research directions.

2. Literature review

There are many papers studying truck scheduling in the cross-dock; yet, they mostly consider the same assumption. Each truck only serves one customer or supplier. But, in the real world, each truck commonly serves more than one supplier (customer) in the pickup (delivery) process. As a result, to have a more realistic cross-docking system, the physical flow from suppliers to the cross-dock (pickup process) and also from the cross-dock to the customers (delivery process) should also be taken into consideration. This would require both truck routing and scheduling in the pickup and delivery processes. In this section, the literature review is presented in two fields, the literature of truck scheduling and also truck routing for the cross-dock.

2.1. The literature review of truck scheduling in the cross-dock

One of the first researches on truck scheduling is published by Yu [26]. In this research, thirty-two different models were identified based on the number of docks available at the site, the dock holding pattern for trucks, and the existence of temporary storage. That research focused on three out of the thirty-two different models. In all the three models, it was assumed that there is only one inbound door and only one outbound door. For each of these models, several solution approaches were developed. The purpose was to find the best truck docking sequence for both inbound and outbound trucks that minimizes the total operation time or maximizes the throughput of the cross-docking system.

Vahdani *et al.* [25] and Soltani and Sadjadi [22] worked on the second model presented by Yu [26] in which there is no temporary storage, and therefore, trucks may not be able to load or unload all of their products at one visit of the unloading dock. Hence, they come back to the dock to continue their tasks in case they could not complete the unloading at the forst visit of the dock. In Vahdani *et al.* [24], two hybrid metaheuristics, called hybrid genetic algorithm and hybrid electromagnetism-like algorithm, were developed. In Soltani and Sadjadi [22], a hybrid simulated annealing and hybrid variable neighborhood search were developed. To obtain robust algorithms, Taguchi method was applied to tune the parameters of each algorithm. Then, these algorithms were compared with the heuristics and tabu search method presented by Yu [26]. The results showed that these algorithms outperform the foregoing algorithms presented by Yu [26].

Li *et al.* [17] proposed a model based on just-in-time scheduling to minimize the storage time and order picking activity. They developed an integer programming model and proposed two metaheuristics based on genetic algorithms to solve the problem. The results showed that these algorithms outperform the model solved by software CPLEX. Alvarez-Perez *et al.* [3] studied this model by developing different metaheuristics called reactive Grasp and tabu search. Their algorithms outperformed those proposed by Li *et al.* [17].

Yu and Egbelu [27] developed a mixed integer programming model for a problem where there is one inbound door, one outbound door and a temporary storage buffer to hold items, located at the shipping dock. The product assignments from inbound trucks to outbound trucks are also determined simultaneously with the sequences of the inbound and outbound trucks. To solve this problem, three different solution approaches were suggested. In the first approach, a mathematical model to solve small size problems was developed. In the second approach, the complete enumeration method was applied to generate all possible truck sequences. In the third approach, a heuristic algorithm was employed to increase solution efficiency.

This research was extended by Boloori Arabani *et al.* [8]. They developed a simulated annealing algorithm to determine the best sequence of inbound and outbound trucks to minimize the makespan. The results showed that this algorithm outperforms the heuristic proposed by Yu and Egbelu [27]. Among the other papers extending Yu and Egbelu's research [27] by applying metaheuristic algorithms, we can refer to Vahdani and Zandieh [23] and Boloori Arabani *et al.* [5]. Furthermore, Boloori Arabani *et al.* [4] extended Yu and Egbelu's research [27] by proposing a scheduling problem of inbound and outbound trailers in a cross-docking system according to just-in-time concept. Three metaheuristics (genetic algorithm, particle swarm optimization and differential evolution algorithm) were presented. Their parameters were set by Taguchi experimental design method. The results showed that the genetic algorithm is the most effective algorithm.

Forouharfard and Zandieh [12] continued Yu and Egbelu's research [27] by presenting an imperialistic competitive algorithm. The objective is to find the best sequence of receiving and shipping trucks to minimize the total number of products. Moreover, Boloori Arabani *et al.* [6, 7] continued Yu and Egbelu's research [27] by considering a cross-docking scheduling problem in which more than one objective is taken into account: minimization of makespan and minimization of the total lateness of outbound trucks. Recently, Mohtashami *et al.* [19] studied the multi objective variant of the problem to minimize both makespan and transportation cost.

2.2. The literature review of truck routing for the cross-dock

Truck routing in the cross-dock was first studied by Lee *et al.* [16]. They proposed an integer nonlinear programming model for a combination of the cross-docking with the pickup and delivery process in the supply chain. They proposed a heuristic algorithm based on a tabu search algorithm to determine the number of vehicles and the best route, schedule, and arrival time of each vehicle at a cross-dock to minimize the transportation cost and the fixed cost of vehicles.

Liao *et al.* [18] considered the same problem and presented a new tabu search algorithm. They evaluated the algorithm by comparing with the earlier tabu search proposed by Lee *et al.* [16]. Vahdani *et al.* [24] also studied the same problem and proposed a new hybrid metaheuristic for vehicle routing scheduling in the cross-dock. This new hybrid algorithm is a combination of particle swarm optimization, simulated annealing and variable neighborhood search. Then, this algorithm was compared with the tabu search algorithm proposed by Lee *et al.* [16]. Santos *et al.* [20] presented an integer programming model and a branch-and-price algorithm for the vehicle routing problem with cross-docking.

Jagannathan [15] developed a mixed integer linear programming model for the vehicle routing problem for cross-docks in order to obtain feasible vehicle routes and schedules. This problem was characterized by heterogeneous vehicles, split deliveries, discrete time windows, line haul, and backhaul operations. This mathematical

TABLE 1. The demands of customers for the example.

Customer	Demand of product type		
	1	2	3
1	21	18	20
2	13	24	19
3	9	9	11
4	8	10	6
5	9	12	2

model was solved by CPLEX. A heuristic approach was also proposed to obtain the initial feasible vehicle routes. Agustina *et al.* [1] studied the vehicle routing problem in cross dock in food supply chain with time windows for delivery. Moghadam *et al.* [21] considered truck routing in the cross dock and developed a mixed integer nonlinear model as well as two metaheuristics for the problem.

Although there are some papers considering the truck routing and scheduling for cross-docking, they rarely take into account truck scheduling at the cross-dock. As they consider no limitation on the number of the doors of the cross dock, they do not determine the sequence of inbound and outbound trucks. This paper deals with a cross-dock with one inbound and one outbound door. Each inbound (outbound) truck can serve more than one supplier (customer) in the pickup (delivery) process. Hence, the vehicle routing scheduling in these processes is also proposed.

3. Problem formulation

This section mathematically formulates the problem of truck routing and scheduling for cross-docking. The routing decisions include the number of required trucks and which inbound (outbound) truck serves which supplier (customer). The scheduling decision includes the product assignments from inbound trucks to outbound trucks and the docking sequences of the inbound and outbound trucks with unlimited temporary storage. The objective of this model is to minimize the makespan with the least number of trucks.

In order to clarify the problem, the assumptions are given as follows.

1. There is one inbound door and one outbound door and they are separate.
2. Each supplier and customer is only visited once by one truck.
3. The inbound and outbound trucks are homogeneous and have unlimited capacity.
4. All inbound and outbound trucks are available at time zero.
5. All inbound and outbound trucks must stay at the doors until they finish their task (*i.e.*, all its loading or unloading activities must be completed).
6. The truck changeover time is the same for all inbound and outbound trucks.
7. Unloading time from an inbound truck and loading time into an outbound truck are the same for all product types and it takes one unit of time for one unit of product.
8. All different products move from the inbound door to the outbound door on a conveyor or system of conveyors in a fixed time.

An illustrative example of the problem is given for better clarification as follows. We have an instance with four suppliers ($p = 4$), five customers ($d = 5$), three types of products ($m = 3$), two inbound trucks ($v_1 = 2$) and two outbound trucks ($v_2 = 2$). The fixed cost of each truck and the cost of each unit of elapsed time are equal to 1 unit of cost. The time for each truck's changeover is considered to be 3 time units and the time consumed to transfer each product via conveyor from inbound door to outbound door is 5 time units. The demands of customers in each type of product are given in Table 1.

In this example, the node number 0 displays the location of the trucks in the cross-dock. The nodes numbers 1 to 4 show the suppliers and the nodes numbers 5 to 9 indicate the customers. The nodes numbers 10 and 11

TABLE 2. The transportation times between nodes for the example.

Node	0	1	2	3	4	5	6	7	8	9	10	11
0		50	80	55	25	–	–	–	–	–	–	0
1	–		35	45	43	–	–	–	–	–	51	–
2	–	35		30	35	–	–	–	–	–	47	–
3	–	45	30		20	–	–	–	–	–	20	–
4	–	43	35	20		–	–	–	–	–	33	–
5	39	–	–	–	–		20	30	35	33	–	–
6	38	–	–	–	–	20		25	30	26	–	–
7	21	–	–	–	–	30	25		10	24	–	–
8	14	–	–	–	–	35	30	10		21	–	–
9	13	–	–	–	–	33	26	24	21		–	–
10	0	–	–	–	–	–	–	–	–	–		–
11	–	–	–	–	–	38	37	20	13	12	–	

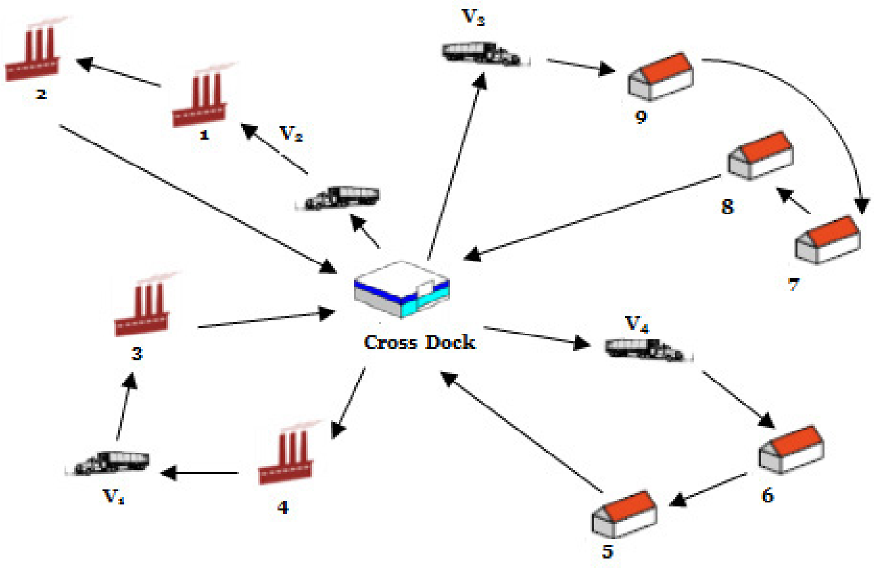

FIGURE 1. The assignment of the suppliers and customers to the trucks for the example.

present the inbound and outbound doors of the cross-dock, respectively. The inbound trucks are shown by V_1 and V_2 and the outbound trucks are shown by V_3 and V_4. The transportation times between nodes are presented in Table 2.

One solution for this example is illustrated in Figures 1–4. The assignment of the suppliers and customers to the trucks are shown in Figure 1. The product assignments from the inbound trucks to the outbound trucks are given in Table 3. Also, the docking sequences of the inbound and outbound trucks and the resulting Gantt charts for this example are shown in Figures 2–4, respectively. Finally, the objective function becomes 534 cost units.

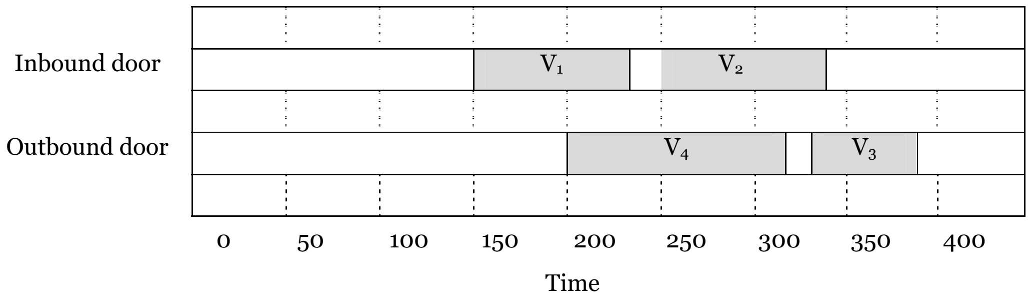

FIGURE 2. The docking sequences of the trucks for the example.

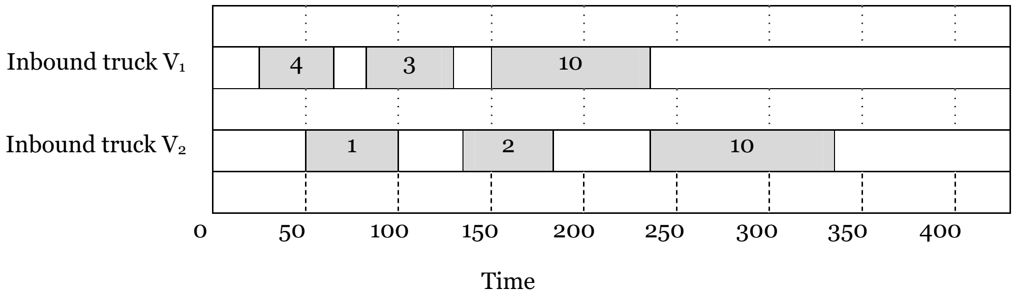

FIGURE 3. Gantt chart of the inbound trucks for the example.

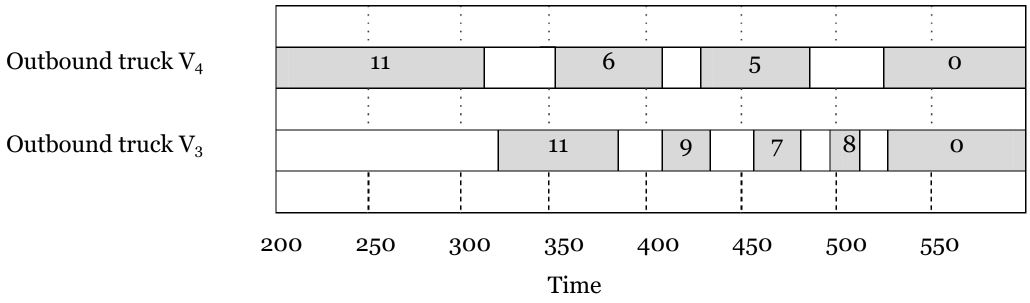

FIGURE 4. Gantt chart of the outbound trucks for the example.

TABLE 3. The product assignments for the example.

From inbound truck	To outbound truck	No. of transferred product types		
		1	2	3
V_1	V_3	–	6	–
V_1	V_4	29	24	26
V_2	V_3	26	25	19
V_2	V_4	5	18	13

The model presented to formulate the problem is in form of a mixed integer linear program. The following notations are used in the model.

o	The location of trucks in the cross dock		
r	The inbound door of the cross dock		
s	The outbound door of the cross dock		
P	Set of suppliers	$(P = 1, 2, \ldots, p)$	$\lvert P\rvert = p$
D	Set of costumers	$(D = 1, 2, \ldots, d)$	$\lvert D\rvert = d$
N	Set of nodes	$(N = 1, 2, \ldots, n)$	$\lvert N\rvert = n = p + d + 3$
M	Set of product types	$(M = 1, 2, \ldots, m)$	$\lvert M\rvert = m$
K_1	Set of inbound trucks	$(K_1 = 1, 2, \ldots, v_1)$	$\lvert K_1\rvert = v_1$
K_2	Set of outbound trucks	$(K_2 = 1, 2, \ldots, v_2)$	$\lvert K_2\rvert = v_2$
K	Set of trucks	$(K = 1, 2, \ldots, v)$	$\lvert K\rvert = v = v_1 + v_2$
f	The fixed cost of using each truck		
c	The cost of each unit of the elapsed time		
g	The truck changeover time		
w	The transportation time of products from the inbound door to the outbound door		
p_{ia}	The number of units of product type a that is loaded from supplier i		
d_{ia}	The number of units of product type a that is delivered to the costumer i		
et_{ij}	The travel time from node i to node j		
h_a	The unloading and loading time for one unit of product type a		
B	A large positive number		

In the model, the following decision variables are defined:

$X_{ij}^{k_1}$	Binary variable taking value 1 if the truck k_1 moves from the node i to the node j in the pickup process, and o otherwise.
$Z_{ij}^{k_2}$	Binary variable taking value 1 if the truck k_2 moves from the node i to the node j in the delivery process, and o otherwise.
Y_{ik}	Binary variable taking value 1 if node i assigns to truck k, and o otherwise.
$T_{k_1k_2}$	Binary variable taking value 1 if any products transfer from inbound truck k_1 to outbound truck k_2, and o otherwise.
$U_{k_1k'_1}$	Binary variable taking value 1 if inbound truck k_1 precedes inbound truck k'_1 in the inbound truck sequence, and o otherwise.
$Q_{k_2k'_2}$	Binary variable taking value 1 if outbound truck k_2 precedes outbound truck k'_2 in the outbound truck sequence, and o otherwise.
DT_{ik}	Continuous variable for the departure time of truck k from node i.
AT_{ik}	Continuous variable for the arrival time of truck k at node i.
S_{ik}	Continuous variable for the time at which truck k starts its operation at node i.
T	Continuous variable for the Makespan.
$C_{k_1k_2a}$	Integer variable for the number of units of product type a transferred from inbound truck k_1 to outbound truck k_2.

The mathematical model for the problem can be described as follows.

$$MIN \qquad c \times T + f \sum_{j \in P} \sum_{k_1 \in K_1} X_{oj}^{k_1} + f \sum_{s \in D} \sum_{k_2 \in K_2} Z_{os}^{k_2} \tag{3.1}$$

Subject to:

$$\sum_{i \in o,P} \sum_{k_1 \in K_1} X_{ij}^{k_1} = 1 \qquad \forall j \in P, \quad i \neq j \tag{3.2}$$

$$\sum_{i \in s,D} \sum_{k_2 \in K_2} Z_{ij}^{k_2} = 1 \qquad \forall j \in D, \quad i \neq j \tag{3.3}$$

$$\sum_{j \in P} \sum_{k_1 \in K_1} X_{oj}^{k_1} \leqslant v_1 \tag{3.4}$$

$$\sum_{k_2 \in K_2} Z_{os}^{k_2} \leqslant v_2 \tag{3.5}$$

$$\sum_{i \in o,P} X_{il}^{k_1} - \sum_{j \in P,r} X_{lj}^{k_1} = 0 \qquad \forall l \in P, \quad \forall k_1 \in K_1, \quad i \neq j \neq l \tag{3.6}$$

$$\sum_{i \in s,D} Z_{il}^{k_2} - \sum_{j \in o,D} Z_{lj}^{k_2} = 0 \qquad \forall l \in D, \quad \forall k_2 \in K_2, \quad i \neq j \neq l \tag{3.7}$$

$$\sum_{i \in P} X_{ir}^{k_1} - X_{ro}^{k_1} = 0 \qquad \forall k_1 \in K_1 \tag{3.8}$$

$$Z_{os}^{k_2} - \sum_{j \in D} Z_{sj}^{k_2} = 0 \qquad \forall k_2 \in K_2 \tag{3.9}$$

$$DT_{jk_1} \geqslant DT_{ik_1} + et_{ij} + \sum_{a \in M} p_{ja} \times h_a - B\left(1 - X_{ij}^{k_1}\right) \qquad \forall j \in P, \forall k_1 \in K_1, \forall i \in o,\ P,\ i \neq j \tag{3.10}$$

$$DT_{jk_2} \geqslant DT_{ik_2} + et_{ij} + \sum_{a \in M} d_{ja} \times h_a - B\left(1 - Z_{ij}^{k_2}\right) \qquad \forall j \in D, \forall k_2 \in K_2, \forall i \in s,\ D\, i \neq j \tag{3.11}$$

$$DT_{rk_1} \geqslant S_{rk_1} - B\left(1 - Y_{rk_1}\right) + \sum_{a \in M} \sum_{k_2 \in K_2} C_{k_1 k_2 a} \times h_a \qquad \forall k_1 \in K_1 \tag{3.12}$$

$$DT_{sk_2} \geqslant S_{sk_2} - B\left(1 - Y_{sk_2}\right) + \sum_{a \in M} \sum_{k_1 \in K_1} C_{k_1 k_2 a} \times h_a \qquad \forall k_2 \in K_2 \tag{3.13}$$

$$AT_{rk_1} \geqslant DT_{ik_1} + et_{ir} - B\left(1 - X_{ir}^{k_1}\right) \qquad \forall i \in P, \forall k_1 \in K_1 \tag{3.14}$$

$$AT_{ok_2} \geqslant DT_{ik_2} + et_{io} - B\left(1 - Z_{io}^{k_2}\right) \qquad \forall i \in D, \forall k_2 \in K_2 \tag{3.15}$$

$$AT_{rk_1} - M\left(1 - Y_{rk_1}\right) \leqslant S_{rk_1} \qquad \forall k_1 \in K_1 \tag{3.16}$$

$$\sum_{k_2 \in K_2} C_{k_1 k_2 a} = \sum_{i \in P} p_{ia} \times Y_{ik_1} \qquad \forall k_1 \in K_1, \forall a \in M \tag{3.17}$$

$$\sum_{k_1 \in K_1} C_{k_1 k_2 a} = \sum_{i \in D} d_{ia} \times Y_{ik_2} \qquad \forall k_2 \in K_2, \forall a \in M \tag{3.18}$$

$$C_{k_1 k_2 a} \leqslant B\left(T_{k_1 k_2}\right) \qquad \forall k_1 \in K_1, \forall k_2 \in K_2, a \in M \tag{3.19}$$

$$S_{rk_1'} + B\left(1 - Y_{rk_1'}\right) \geqslant DT_{rk_1} + g - B\left(1 - U_{k_1 k_1'}\right) - B\left(1 - Y_{rk_1}\right) \qquad \forall k_1,\ k_1{}' \in K_1,\ k_1 \neq k_1' \tag{3.20}$$

$$S_{rk_1} + B\left(1 - Y_{rk_1}\right) \geqslant DT_{rk_1'} + g - B\left(U_{k_1k_1'}\right) - B\left(1 - Y_{rk_1'}\right) \qquad \forall k_1,\, k_1{}' \in K_1,\, k_1 \neq k_1' \quad (3.21)$$

$$S_{sk_2'} \geqslant DT_{sk_2} + g - B\left(1 - Q_{k_2k_2'}\right) - B\left(1 - Y_{sk_2}\right) - B\left(1 - Y_{sk_2'}\right) \qquad \forall k_2,\, k_2{}' \in K_2,\, k_2 \neq k_2' \quad (3.22)$$

$$S_{sk_2} \geqslant DT_{sk_2'} + g - B\left(Q_{k_2k_2'}\right) - B\left(1 - Y_{sk_2}\right) - B(1 - Y_{sk_2'}) \qquad \forall k_2, k_2' \in K_2, k_2 \neq k_2' \quad (3.23)$$

$$DT_{sk_2} + B\left(1 - Y_{sk_2}\right) \geqslant S_{rk_1} + w + \sum_{a \in M} C_{k_1k_2a} \times h_a - B\left(1 - T_{k_1k_2}\right) - B\left(1 - Y_{rk_1}\right) \qquad \forall k_1 \in K_1, \forall k_2 \in K_2 \quad (3.24)$$

$$Y_{rk_1} = \sum_{j \in P} X_{oj}^{k_1} \qquad \forall k_1 \in K_1 \quad (3.25)$$

$$Y_{sk_2} = Z_{os}^{k_2} \qquad \forall k_2 \in K_2 \quad (3.26)$$

$$Y_{ik_1} = \sum_{j \in o,P} X_{ji}^{k_1} \qquad \forall i \in P,\, \forall k_1 \in K_1, \quad i \neq j \quad (3.27)$$

$$Y_{ik_2} = \sum_{j \in D,s} Z_{ji}^{k_2} \qquad \forall i \in D,\ \forall k_2 \in K_2,\ i \neq j \quad (3.28)$$

$$T \geqslant AT_{ok_2} \qquad \forall k_2 \in K_2 \quad (3.29)$$

$$X_{ij}^{k_1}, \quad Z_{ij}^{k_2}, \quad Y_{ik}, \quad T_{k_1k_2}, \quad U_{k_1k_1'}, \quad Q_{k_2k_2'} \in \{0, 1\} \qquad \forall\, i, j, kk_1,\, k_1{}',\, k_2,\, k_2' \quad (3.30)$$

$$DT_{ik}, \quad AT_{ik}, \quad S_{ik} \geqslant 0. \qquad \forall\, i,\, k \quad (3.31)$$

Objective function (3.1) minimizes makespan with the least number of trucks used. Constraint sets (3.2) and (3.3) ensure that each supplier or costumer is only served by one truck. Constraint sets (3.4) and (3.5) show that the number of trucks that leave the cross-dock must be less than the number of available trucks. Constraint sets (3.6) to (3.9) express the consecutive movement of trucks. Constraint sets (3.10) to (3.13) express the departure time of a truck from a node. Constraint sets (3.14) and (3.15) calculate the arrival time for each inbound and outbound node, respectively. Constraint set (3.16) connects the arrival time of an inbound truck at the inbound door to the starting time of its operation.

Constraint set (3.17) ensures that the total number of units of product type "a" that transfer from inbound truck "k_1" to all outbound trucks is exactly the same as the number of that product type that is loaded from suppliers by inbound truck "k_1". Similarly, Constraint set (3.18) ensures that the total number of product type "a" that is transferred from all inbound trucks to outbound truck "k_2" is exactly the same as the number of that product type required for outbound truck "k_2" to be delivered to the costumers. Constraint set (3.19) just enforces the correct relationship between the $C_{k_1k_2a}$ variables and the $T_{k_1k_2}$ variables. Constraint sets (3.20) and (3.21) make a valid sequence for starting and completing times for the inbound trucks' operation based on their order.

Similar to constraint sets (3.20) and (3.21), constraint sets (3.22) and (23) function for the outbound trucks. Constraint set (3.24) connects the leaving time for an outbound truck to the starting time of an inbound truck's operation if any products or items are transferred between each pair of trucks. The assignment of trucks to the inbound and outbound doors is shown by constraint sets (3.25) and (3.26), respectively. The assignment of the suppliers and the customers to the trucks are shown by constraint sets (3.27) and (3.28), respectively. Constraint (3.29) sets makespan greater than or equal to the time at which the last scheduled outbound truck arrives at the location of the trucks in the cross-dock in the end of the delivery process. Constraint sets (3.30) and (3.31) specify the domains of the decision variables.

4. The proposed metaheuristics

Since the problem under consideration is NP-hard, metaheuristics are known to be the best algorithms to obtain a good solution in a reasonable amount of time. This paper develops two metaheuristics of generational

12	1	2	5	6	-	-
13	1	4	3	6	-	-
15	7	10	8	1	-	-
14	7	11	9	1	-	-

Figure 5. An example of the solution representation.

genetic algorithm and reactive tabu search to solve the problem. The representation scheme of solutions is the most important part in designing metaheuristics because it can greatly affect the performance of the algorithm. In this paper, the representation scheme is a matrix with $(v_1 + v_2) \times (\max\{d, p\} + 3)$ size. Each row shows the route assigned to a truck. The first column of each row represents the name of the trucks and the rest of the columns represent the nodes assigned to that truck. Also, the sequencing of the rows shows the sequences of the trucks at the cross-dock's doors. An example of the solution representation is shown in Figure 5. Number 1 displays the location of the trucks in the cross-dock. Numbers 2 to 5 show the suppliers and numbers 8 to 11 show the customers. Numbers 6 and 7 present the inbound and outbound door of the cross-dock, respectively. Numbers 12 and 13 indicate the inbound trucks and numbers 14 and 15 indicate the outbound trucks.

Therefore, this scheme shows the assignment of the suppliers to the inbound trucks, the assignment of the customers to the outbound trucks, and also their sequences in the related trucks. Moreover, the sequencing of the trucks at the cross-dock's doors is illustrated in this scheme.

4.1. The genetic algorithm

The genetic algorithm (GA) is a population based algorithm to solve the optimization problems. It means instead of working on one solution in each iteration, it works on a population of solutions and each element of this population is called chromosome. Thus, each chromosome represents a solution of the problem. The chromosomes evolve through successive iterations, called generations. During each generation, the chromosomes are evaluated using some measures of fitness. The overall framework of our applied GA is described as follows.

First, an initial population, a set of chromosomes, is generated by the initialization procedure. For fitness evaluation, the objective function for each chromosome is calculated. Then for obtaining a new chromosome, a selection mechanism, called roulette wheel selection, picks two chromosomes of the current population in a way that it gives a higher chance to better chromosomes. Then, the selected chromosomes are combined by the crossover operation with probability of P_{c}. Afterwards, the mutation operation is carried out on the new chromosome with probability of P_{m}. After selection, crossover and mutation, we apply a local search method to the new chromosome with probability of P_{ls} (P_{c}, P_{m} and P_{ls} are the probabilities of applying the crossover, mutation and local search schemes).

In the proposed genetic algorithm, the new chromosome is accepted to enter the population only if it is better than the worst chromosome of the population and if at the same time unique, *i.e.*, there are no other identical chromosomes already in the population. Otherwise it is rejected. These schemes are repeated for Pop size times and a new generation is obtained. After each generation, the local search method is also applied to the best chromosome by a probability of $2P_{\mathrm{ls}}$.

```
Create an initial population of chromosomes
While stopping criterion is not met repeat
        Counter=0
        For acquiring each new offspring population do
        While the Counter is not equal to Pop size repeat
            -  Calculate the objective function for each chromosome in the population
            -  Apply the selection operation and specify the related parents for conducting
               the crossover scheme based on the $P_c$
            -  Apply the mutation scheme for the new chromosomes based on the $P_m$
            -  Apply the local search method to the new chromosomes according to the $P_{ls}$
            If the acquired chromosomes are better than the worst chromosome of the
            population and
            If at the same time unique then
                They are replaced with the worst chromosome of the population
            Else
                They are rejected.
            End if
            Counter=Counter+2
          End while
          Apply the local search method to the best chromosome by a probability of $2P_{ls}$
        End for
        Apply the restart mechanism whenever the fitness value in the population does not
        change for $[0.015*(p+d)*(v_1+v_2)]+m$ generations
End while
```

Figure 6. The pseudo-code of the proposed genetic algorithm.

This scheme repeats so long as a related stopping criterion is met. During search, the population may achieve a sufficiently low diversity for the process and stall around a local optimum. To overcome this issue, we apply a restart mechanism as an additional phase. Whenever the best solution of the population is not improved for a given number of generations, this mechanism is implemented.

The pseudo-code of the GA is illustrated in Figure 6. In the following, we explain the population initialization procedure and the four main schemes, crossover, mutation, local search and restart briefly.

4.1.1. Population initialization

The first member of the population is generated by the following procedure. One of the suppliers and customers are selected at random and assigned to the first inbound and outbound trucks, respectively. Then the remaining suppliers and customers are selected randomly and inserted in the position that results in the lowest objective function value. So long as the suppliers and customers are inserted into inbound and outbound trucks' position, the solution is feasible. This process is continued until all the suppliers and customers are positioned. Afterwards, the remaining members of the population are generated at random.

4.1.2. Crossover scheme

The crossover operation is actually the combination of two chromosomes in order to create two new chromosomes. For applying the crossover scheme on the selected chromosomes which are chosen by the roulette wheel selection, first two rows are selected randomly from the both related parent chromosomes. Then, the chosen rows of the second parent chromosome plus the rows between these two rows are added to the beginning of the first chosen row of the first parent chromosome. As a result, some suppliers or customers may appear in the acquired chromosome twice. Also, two different routes may be assigned to the same truck. To resolve the conflicts mentioned above, remove the route that originally belongs to the first parent.

If afterwards some suppliers or customers still appear twice, they will be eliminated from the truck that originally belongs to the first parent. After this step, some suppliers or customers may remain unassigned. To this purpose, of all the feasible places for reinserting each of them, the best place (based on fitness value) is chosen and then they are reinserted. As a result of these steps, a complete offspring is obtained and the second offspring is generated by repeating these steps with reversed roles of the parents. Thus, in this crossover scheme, in addition to the sequences of the trucks at the doors of the cross-dock, the places of some suppliers or customers are also changed in the same or in the other route.

To better illustrate the crossover, it is applied to an example. Consider a problem with 4 suppliers (shown by numbers 2 to 5), 4 customers (shown by numbers 8 to 11), 3 inbound trucks (shown by numbers 12 to 14) and 2 outbound trucks (shown by numbers 15 and 16). Therefore, each solution has 5 rows (3 for inbound and 2 for outbound trucks). Each row shows the nodes assigned to the corresponding trucks and also the sequence they are visited. Figure 7 shows the crossover applied to this example. The two rows of outbound trucks 15 and 16 from the first parent and the two rows of inbound truck 12 and the outbound truck 16 from the second parent are selected (Parts 1, 2 and 3 in Fig. 7). The combined solution is infeasible since Suppliers 3, 4 and 5 and Customers 10 and 11 are assigned to more than one route. Moreover, two different routes are assigned to the trucks 12, 14 and 16. To remove these issues, Suppliers 3 and 4 are eliminated from the Truck 13 and Customer 10 is eliminated from Truck 15. Supplier 2 and Customer 9 are reassigned to have a feasible solution (Parts 4 and 5 in Fig. 7).

4.1.3. Mutation scheme

The mutation operator is usually used to keep an acceptable level of diversity in the population. Actually, the mutation operator searches for a solution space which is not found by the crossover operator. In this paper, in order to apply the mutation scheme, first a row related to the inbound trucks and a row related to the outbound trucks are randomly selected from the related chromosome. And their places are changed randomly in the sequencing of the inbound and outbound trucks, respectively. In the next step, a sub-route of the trucks is chosen randomly from the related chromosome. Then, each of the suppliers or customers in this sub-route is reinserted in the best place (based on fitness value) from all the feasible places for reinserting each of them. Hence, in this mutation scheme, the places of some suppliers or customers and also the truck sequences at the cross-dock's doors are changed.

Let us further illustrate the mutation by applying it a numerical example. Consider a problem with 4 suppliers (shown by numbers 2 to 5), 4 customers (shown by numbers 8 to 11), 2 inbound trucks (shown by numbers 12 to 13) and 2 outbound trucks (shown by numbers 14 and 15). Therefore, each solution has 4 rows (2 for inbound and 2 for outbound trucks). The procedure is shown in Figure 8. Suppose Rows related to inbound truck 12 and outbound truck 15 are selected (Parts 1 and 2 in Fig. 8). And, the sub-route includes suppliers 3, 4 and 5. They are reinserted into all position and the best position is selected (Parts 3 and 4 in Fig. 8).

4.1.4. Local search scheme

To enhance the algorithm's performance, GA is hybridized with a local search. The procedure of this local search can be described as follows. The local search method generates some neighborhood solutions by using a neighborhood structure. The neighborhood structure involves two neighborhood operators, named moving and swapping operators. In moving operator, one row that is related to the inbound trucks (outbound trucks) and one supplier (customer) are moved from their current place to another place. In the swapping operator, the places of two rows that are related to the inbound trucks (outbound trucks) and also the places of two suppliers (customers) are swapped. One of these two neighborhood operators is selected randomly. Figure 9 shows a numerical example for local search. Suppose swapping operator is applied to the rows related to the inbound trucks (inbound trucks 13 and 14) and then to the suppliers. The places of all pairs of suppliers are swapped and then the best one is chosen (suppliers 2 and 4). In the next step, suppose the moving operator

12	1	5	6	-	-	-
13	1	4	3	6	-	-
14	1	2	6	-	-	-
15	7	10	8	1	-	-
16	7	11	9	1	-	-

First parent

13	1	2	6	-	-	-
12	1	4	6	-	-	-
14	1	5	3	6	-	-
16	7	10	11	1	-	-
15	7	8	9	1	-	-

Second parent

(1)

12	1	5	6	-	-	-
13	1	4	3	6	-	-
14	1	2	6	-	-	-
12	1	4	6	-	-	-
14	1	5	3	6	-	-
16	7	10	11	1	-	-
15	7	10	8	1	-	-
16	7	11	9	1	-	-

(2)

13	1	4	3	6	-	-
12	1	4	6	-	-	-
14	1	5	3	6	-	-
16	7	10	11	1	-	-
15	7	10	8	1	-	-

2	9

(3)

13		1	4	3	6	-	-
12		1	4	6	-	-	-
14		1	5	3	6	-	-
16		7	10	11	1	-	-
15		7	10	8	1	-	-

2	9

(4)

13	1	2	6	-	-	-
12	1	4	6	-	-	-
14	1	5	3	6	-	-
16	7	10	11	1	-	-
15	7	9	8	1	-	-

(5)

Number 1	Location of the trucks in the cross dock
Numbers 2 to 5	Suppliers
Number 6	Inbound door
Number 7	Outbound door
Numbers 8 to 11	customers
Numbers 12 to 14	Inbound trucks
Numbers 15 and 16	Outbound trucks

Figure 7. An example of the crossover scheme.

is then applied to the rows related to the outbound trucks (outbound truck 16) and then to the customers. Each customer is removed from its current place and reinserted into all the feasible places. Finally, the best move is chosen (customer 8).

It is also needed to mention that by using the moving operator, the selected supplier or customer is enable to insert in the same or in the other route. Also, by using the swapping operator, the possibility of swapping two suppliers or two customers in one or two routes is allowed. Therefore, in this structure, the places of some suppliers and customers and also the truck sequences at the cross-dock's doors are changed. The local search

13	1	2	6	-	-	-
12	1	3	4	5	6	-
14	7	10	11	1	-	-
15	7	8	9	1	-	-

(1)

12	1	3	4	5	6	-
13	1	2	6	-	-	-
15	7	8	9	1	-	-
14	7	10	11	1	-	-

(2)

12	1	3	4	5	6	-
13	1	2	6	-	-	-
15	7	8	9	1	-	-
14	7	10	11	1	-	-

(3)

12	1	4	6	-	-	-
13	1	3	2	5	6	-
15	7	10	11	1	-	-
14	7	8	9	1	-	-

(4)

Number 1	Location of the trucks in the cross dock
Numbers 2 to 5	Suppliers
Number 6	Inbound door
Number 7	Outbound door
Numbers 8 to 11	customers
Numbers 12 and 13	Inbound trucks
Numbers 14 and 15	Outbound trucks

FIGURE 8. An example of the mutation scheme.

phase stops when a local optimum is found, *i.e.* if after having examined all the neighbors, no improvements are found.

4.1.5. Restart scheme

In this paper, every time the fitness value in the population does not change for more than

$$[0.015 \times (p+d) \times (v_1+v_2)] + m. \tag{4.1}$$

Generations in row, the restart procedure will keep the 20% best chromosomes from the current population and regenerate the remaining 80% at random, hoping to reintroduce diversity in the population and to escape from local optimum.

4.2. Reactive tabu search algorithm

The tabu search algorithm is a metaheuristic algorithm which was first introduced by Glover [14]. The tabu search approach like other metaheuristic approaches operates on this basic assumption that for each obtained solution we can generate some neighborhood solutions. The objective of tabu Search is to avoid cycles while applying a local search technique. In the Reactive tabu Search, the tabu tenure (tabu list size) is adapted to further consider this issue.

To make the tabu search algorithm reactive, we must evaluate the need of diversification of the search. When the same solution is explored twice, the search must be diversified. In order to detect redundancies, we memorize

14	1	5	2	6	-	-	-
12	1	3	6	-	-	-	-
13	1	4	6	-	-	-	-
15	7	10	8	11	1	-	-
17	7	9	1	-	-	-	-
16	7	12	1	-	-	-	-

13	1	4	6	-	-	-	-
12	1	3	6	-	-	-	-
14	1	5	2	6	-	-	-
15	7	10	8	11	1	-	-
17	7	9	1	-	-	-	-
16	7	12	1	-	-	-	

(1)

13	1	4	6	-	-	-	-
12	1	3	6	-	-	-	-
14	1	5	2	6	-	-	-
15	7	10	8	11	1	-	-
17	7	9	1	-	-	-	-
16	7	12	1	-	-	-	

13	1	2	6	-	-	-	-
12	1	3	6	-	-	-	-
14	1	5	4	6	-	-	-
15	7	10	8	11	1	-	-
17	7	9	1	-	-	-	-
16	7	12	1	-	-	-	

(2)

13	1	2	6	-	-	-	-
12	1	3	6	-	-	-	-
14	1	5	4	6	-	-	-
15	7	10	8	11	1	-	-
17	7	9	1	-	-	-	-
16	7	12	1	-	-	-	

13	1	2	6	-	-	-	-
12	1	3	6	-	-	-	-
14	1	5	4	6	-	-	-
16	7	12	1	-	-	-	-
15	7	10	8	11	1	-	-
17	7	9	1	-	-	-	-

(3)

13	1	2	6	-	-	-	-
12	1	3	6	-	-	-	-
14	1	5	4	6	-	-	-
16	7	12	1	-	-	-	-
15	7	10	8	11	1	-	-
17	7	9	1	-	-	-	-

13	1	2	6	-	-	-	-
12	1	3	6	-	-	-	-
14	1	5	4	6	-	-	-
16	7	12	8	1	-	-	-
15	7	10	11	1	-	-	-
17	7	9	1	-	-	-	-

(4)

Number 1	Location of the trucks in the cross dock
Numbers 2 to 5	Suppliers
Number 6	Inbound door
Number 7	Outbound door
Numbers 8 to 12	customers
Numbers 12 to 14	Inbound trucks
Numbers 15 to 17	Outbound trucks

FIGURE 9. An example of the neighborhood structure.

each explored solution. If we have a repeated solution, the list length is increased. On the contrary, when there is no repeated solution during a fixed number of iterations, thus indicating that search is diversified enough, we can reduce the list length.

In this paper, we present a reactive tabu search algorithm with path relinking technique. To reach the optimum solution in the proposed problem, the reactive tabu search algorithm starts from an initial solution by the procedure described in Section 4.1.1. Then, the algorithm generates some neighborhood solutions using the proposed neighborhood structure described in Section 4.1.4. To determine the solution value, the objective function for each neighborhood solution is calculated. After that, the algorithm chooses the best solution among the current neighborhood solutions and checks the tabu list.

If the related move is not a tabu move, or if it is a tabu move but it satisfies the Aspiration criterion (if the neighborhood solution is better than the best obtained solution until now), then the solution is chosen by the algorithm. Otherwise, pick the best move that is not a tabu move and its solution is considered as a new current solution. After choosing the neighborhood solution, the tabu list is updated. It means that, the previous move by which we chose the neighborhood solution is stored in the tabu list to prevent the algorithm from returning back to the related solution and creating a cycle.

After storing the previous move in the tabu list, some of the moves that were stored in the tabu list before, exit from the list. The time that the moves are stored in the tabu list is called tabu tenure. Moving from the current solution to the neighborhood solution is continued until the stopping criterion is met. As stopping criterion for the proposed reactive tabu search algorithm, computational time is employed.

The advantage of keeping of all visited solutions and using the reactive scheme is to avoid that the search be confined within small regions of the solution space. To overcome this ineffective behavior, we enforce a diversification step whenever the same solution is visited by the search more than a certain number (Max repetition) of times. The same procedure described above for the construction of initial solution is used to generate a new solution to restart the search at a diversification step.

In this paper, every time the solution value does not change for a number of iterations (equal to Eq. (4.1)), the path relinking procedure is applied. The objective is to generate a set of new solutions in between the good solutions and then the best one is selected as a new current solution.

Generally, in the proposed reactive tabu search, to escape from a local optimum, there is a possibility of accepting worse solutions than the best obtained solution. In order to prevent creating a cycle, the some previous moves are stored in the tabu list and it is banned to choose them. In particular, intensification and diversification are balanced by adjusting the tenure of the tabu list during the search. The pseudo-code of the RTS is illustrated in Figure 10. In the following, we explain the tabu tenure adjustment and the path relinking mechanism in the RTS briefly.

4.2.1. Tabu tenure adjustment mechanism

The length of the tabu list is a crucial parameter in the performance of tabu search since it controls the amount of intensification and diversification capability. Hence, the reactive tabu search algorithm designs a mechanism to maintain the length of the tabu list variable in order to avoid previously visited solutions. The procedure of this tabu tenure adjustment can be described as follows.

The initial tabu tenure is set to 10. There is a counter rep(S) associated with each solution S, representing the number of times the related solution is visited. The tabu tenure is multiplied by a factor ($P1$) whenever the algorithm faces a previously visited solution. The tabu tenure is divided by the same factor whenever all moves turn out to be forbidden. The tabu tenure is also gradually and periodically reduced, in order to make the search less restrictive in regions less subjected to cycling (in our implementation the tabu tenure is divided by $P2$ after each group of n iterations without any solution previously visited).

4.2.2. Path relinking

Path relinking is a search technique originally proposed by Glover and Laguna [14] where the objective is to explore the search space or "path" between a given set (usually two) of good solutions. The path relinking

```
Create an initial solution
X=initial solution
Best Obj=Obj(X)
Tabu tenure=10
Counter1=0
Counter2=0
While stopping criterion is not met repeat
    Generate new neighborhood solutions and find the best solution
    X=the best solution
    Update the tabu list with the best moves
    If all moves turn out to be forbidden then
        Tabu tenure=tabu tenure/P1
    End if

    If the current solution (X) is a repeated solution then
        Counter1=0
        rep(S)=rep(S)+1
        If rep(S)>=Max repetition then
            Execute the diversification step
            X=new solution at the diversification step
        Else
            Tabu teure=tabu tenure*P1
        End if
    Else
            Counter1=Counter1+1
            If Counter1=n then
                Tabu tenure=tabu tenure/P2
                Counter1=0
            End if
    End if

    If Obj(X)<Best Obj then
        Best Obj=Obj(X)
        Counter2=0
    Else
        Counter2= Counter2+1
        If Counter2=$[0.015*(p+d)*(v_1+v_2)]+m$ then
            Apply the path relinking procedure
            Counter2=0
            X= new solution at the path relinking step
            If Obj(X)<Best Obj then
                Best Obj=Obj(X)
            End if
        End if
    End if
End while
```

FIGURE 10. The pseudo-code of the proposed reactive tabu search algorithm.

technique is applied after a number of iterations (equal to Eq. (4.1)) without improvement in the best solution as follows.

In this paper, to make the selected solution (called origin) more similar to another solution (called destination), all the suppliers (customers) swaps (*i.e.*, suppliers interchange) are determined. Then, these swaps are implemented on the origin. By implementing each change, the origin becomes more similar to the destination. For each intermediate solution, the objective value is obtained and at the end the intermediate solution with the lowest objective value among all the movements is returned as a new current solution.

With respect to the selection of the two individuals to carry out the path relinking, we select the two best unmarked solutions from a pool of elite solutions. The selected solutions are marked in order to avoid selecting

TABLE 4. The parameters and their levels.

Genetic Algorithm		Reactive tabu search Algorithm	
Parameter	Levels	Parameter	Levels
Population size (Pop size)	20, 30, 40	$P1$	1.1, 1.2, 1.3
Crossover probability (P_c)	0.7, 0.8, 0.9	$P2$	1.1, 1.2, 1.3
Mutation probability (P_m)	0.01, 0.05, 0.09	Max repetition	8, 10, 12
Local search Probability (P_ls)	0.1, 0.15, 0.2	Number of iteration without repetition (n)	8, 10, 12

them again with each other for the path relinking. As a move, we use an operator similar to mutation already explained as mutation in GA.

5. Parameter setting

To have a more effective algorithm, the parameters of the proposed algorithms are tuned. To this end, an experiment is conducted to determine the best level of parameters. We use the relative percentage deviation (RPD) as a performance measure to compare the parameters' levels of each algorithm. It is calculated as follows.

$$\mathrm{RPD} = \frac{Alg_{sol} - Min_{sol}}{Min_{sol}} \times 100$$

where Alg_{sol} is the solution obtained by an algorithm for a given instance, and Min_{sol} is the best solution known for that instance.

The parameters of the proposed algorithms and their levels are shown in Table 4.

With these levels, for GA and RTS, there are 81 treatments. 24 instances are randomly generated as follows:

$$\begin{aligned}(p, d) &= \{(10, 10), (15, 15), (20, 20)\}\\ (v_1, v_2) &= \{(10, 10), (20, 20)\}\\ m &= \{5, 10\}.\end{aligned}$$

For each of 12 above combinations, two replications are performed. The transportation times between nodes are generated from a uniform distribution over [20,100]. The fixed cost of each truck and the cost of each unit of elapsed time are equal to 1000 and 1 units of cost, respectively. The time for each truck's changeover is considered to be 75 time units and the time consumed to transfer each product via conveyor from inbound door to outbound door is 100 time units. The total number of products is also equal to 2000 product units. The generated instances are available upon request to the authors.

To execute the experiments, the algorithms are coded in C++ and run on a notebook with 1.83 GHz Intel® Core™ i2 CPU T5600 and 3 GB of RAM memory. Tables 5 and 6 show the results of GA and RTS, respectively. In GA, the population size of 40, crossover and mutation probability of 0.7 and 0.09, and local search probability of 0.15 obtain the lowest average RPD. In RTS, $P1$ and $P2$ of 1.1, max repetition of 8 and iteration without repetition of 12 outperform the other levels.

6. Numerical evaluation

This section evaluates the proposed metaheuristics for performance. To this end, two experiments are performed. In the first one, a set of small instances is generated and the metaheuristics are compared against the optimal solution obtained by the mathematical model. In the second experiment, metaheuristics are further evaluated on large instances. The stopping criterion for the proposed algorithms is a fixed computational time. In this case, we can have a fair comparison among algorithms by giving them the same computational time.

Table 5. The average of RPD obtained in each level of parameters of the genetic algorithm.

$p-d/v_1-v_2/m$	Population size			Crossover probability (P_c)			Mutation probability (P_m)			Local search probability (P_{ls})		
	20	30	40	0.7	0.8	0.9	0.01	0.05	0.09	0.1	0.15	0.2
10-10/10-10/5	0.23	0.00	0.02	0.03	0.20	0.17	0.27	0.00	0.03	0.05	0.28	0.16
10-10/10-10/10	0.28	0.00	0.00	0.00	0.37	0.23	0.05	0.15	0.03	0.41	0.06	0.00
10-10/20-20/5	0.32	0.01	0.10	0.02	2.59	0.17	0.74	0.22	0.00	0.19	0.09	0.00
10-10/20-20/10	0.12	0.01	0.03	0.14	0.26	0.28	0.27	0.19	0.37	0.11	0.29	0.11
15-15/10-10/5	0.00	0.09	0.13	0.13	0.31	0.21	0.24	0.00	2.64	4.99	0.10	1.73
15-15/10-10/10	0.19	0.01	0.28	2.48	0.00	2.54	0.88	0.31	0.00	0.67	0.16	0.00
15-15/20-20/5	3.05	0.00	0.16	0.00	2.77	3.27	0.12	5.86	2.16	0.52	0.00	2.96
15-15/20-20/10	0.57	2.83	0.00	0.08	0.59	2.87	2.29	2.49	1.87	2.51	0.06	2.73
20-20/10-10/5	0.04	0.19	0.19	0.32	0.18	2.43	0.69	2.29	0.29	2.83	0.00	0.04
20-20/10-10/10	0.57	0.47	0	0.23	2.56	2.79	2.77	0.26	0.17	0.63	0.18	2.61
20-20/20-20/5	2.76	0.18	0.04	0.30	2.66	3.53	4.61	6.97	0.22	3.41	2.56	2.42
20-20/20-20/10	1.78	0.17	1.79	2.37	2.58	2.54	2.84	2.80	2.65	1.87	4.06	2.93
Average	0.82	0.33	0.22	0.50	1.25	1.75	1.31	1.79	0.86	1.51	0.65	1.30

Table 6. The average of RPD obtained in each level of parameters of the reactive tabu search.

$p-d/v_1-v_2/m$	$P1$			$P2$			Max repetition			Number of iteration without repetition (n)		
	1.1	1.2	1.3	1.1	1.2	1.3	8	10	12	8	10	12
10-10/10-10/5	0.03	0.11	0.03	0.23	0.13	0.08	0.02	0.01	0.08	0.09	0.08	0.22
10-10/10-10/10	0.05	0.18	0.22	0.17	0.00	0.08	0.08	0.08	0.25	0.09	0.34	0.00
10-10/20-20/5	0.06	0.12	0.15	0.13	0.01	0.03	0.07	0.00	0.18	0.19	0.19	0.03
10-10/20-20/10	0.29	0.10	0.00	0.00	0.18	0.26	0.29	0.29	0.00	0.05	0.20	0.16
15-15/10-10/5	0.02	0.34	0.17	0.29	0.04	0.31	0.00	0.67	0.55	0.32	0.23	0.03
15-15/10-10/10	0.18	0.11	0.19	0.00	0.36	0.26	0.03	0.48	0.01	0.34	0.00	0.13
15-15/20-20/5	0.33	0.23	0.00	0.13	0.06	0.18	0.00	0.25	0.22	0.05	0.00	0.21
15-15/20-20/10	0.04	0.06	0.44	0.08	0.26	0.37	0.05	0.29	0.30	0.01	0.21	0.01
20-20/10-10/5	0.12	0.00	0.14	0.22	0.38	0.52	0.53	0.16	0.16	0.03	0.27	0.16
20-20/10-10/10	0.15	0.07	0.19	0.37	0.34	0.00	0.28	0.15	0.37	0.09	0.08	0.03
20-20/20-20/5	0.00	0.06	0.27	0.12	0.04	0.00	0.11	0.15	0.29	0.00	0.19	0.29
20-20/20-20/10	0.37	0.48	0.04	0.00	0.11	0.22	0.50	0.53	0.00	0.29	0.16	0.08
Average	0.13	0.15	0.15	0.14	0.15	0.19	0.16	0.25	0.20	0.12	0.16	0.11

6.1. Experiment with small instances

In this subsection, a set of small instances are generated and solved by the mathematical model. We use CPLEX 10.1 software. We consider two inbound and two outbound trucks. The numbers of product types are 2, 3 and 4. The other characteristics of these instances are as follows. The transportation times between nodes are generated from a uniform distribution over [20,200]. The fixed cost of each truck and the cost of each unit of elapsed time are equal to 1000 and 1 units of cost, respectively. The time for each truck's changeover is considered to be 75 time units and the time consumed to transfer each product via conveyor from inbound door to outbound door is 100 time units. The total number of products is also equal to 500 product units.

Table 7 shows the results and optimality gap of the algorithms. The instances with 3 suppliers and 3 customers are optimally solved in less than one second while the instances with 5 suppliers and 5 customers or less are solved in less than 100 s. The model is capable of optimally solving instances up to 5 suppliers and 6 customers in 1022 s. The proposed metaheuristics solve all the small instances to optimality.

Table 7. The results of the algorithms for the small problems.

Instance		Model		GA	RTS
No. of suppliers	No. of customers	Average objective value	Average CPU time (s)	Average optimality gap	Average optimality gap
2	2	4165.33	0.03	0	0
2	3	4066.33	0.09	0	0
3	3	4218.66	0.15	0	0
3	4	4282.00	1.01	0	0
4	4	4339.33	2.86	0	0
4	5	4319.00	11.60	0	0
5	5	4435.33	52.67	0	0
5	6	4481.00	1022.83	0	0

Table 8. The average RPD of the proposed algorithms versus $(p - d)$.

$p - d$	Genetic	Reactive tabu search
5–5	0.00	0.00
5–10	0.06	0.01
10–5	0.54	0.00
10–10	0.13	0.02
10–15	0.12	3.15
15–10	0.42	1.06
15–15	0.04	1.78
15–20	0.00	8.57
20–15	0.00	4.78
20–20	0.00	9.29
Average	0.13	2.86

6.2. Experiment with large instances

In order to compare the performance of these proposed metaheuristics, a set of 60 large instances are also generated. The characteristics of these instances are as follows. The transportation times between nodes in the first and second instances are generated from a uniform distribution in the range [20,200] and [20,100], respectively. The fixed cost of each truck and the cost of each unit of elapsed time are equal to 1000 and 1 units of cost, respectively. The time for each truck's changeover in the first and second instances is considered to be 75 and 80 time units, respectively. And the time consumed to transfer each product via conveyor from inbound door to outbound door in the first and second instances is 100 and 150 time units, respectively. The total number of products in the first and second instances is also equal to 2000 and 3000 product units, respectively. For the combination of the parameters, we generate two different instances. Therefore, it sums up to 120 instances.

As mentioned in Section 4, the used stopping criterion for the algorithms is a fixed CPU time depending on the problem size. Hence, the stopping criterion is set to as a fixed computational time of $0.2 \times [(p \times d) + (v_1 * v_2)] \times m$ s.

Table 8 shows the obtained average relative percentage deviation for the proposed algorithms versus $(p - d)$. A clear trend here is that RTS works better in small instances while GA is better in large instances. Moreover, Figure 11 shows the performance of the algorithms versus the number of suppliers and customers. The results show that as the number of suppliers (customers) increases, the performance of RTS becomes worse. The performance of the genetic algorithm is robust. Note that in instances with fewer suppliers and customers, RTS outperforms GA. Yet, after $p - d$ of 5, GA becomes better. The difference in performance becomes more significant in larger sizes. The reason likely comes from this fact that RTS is a local search based algorithm

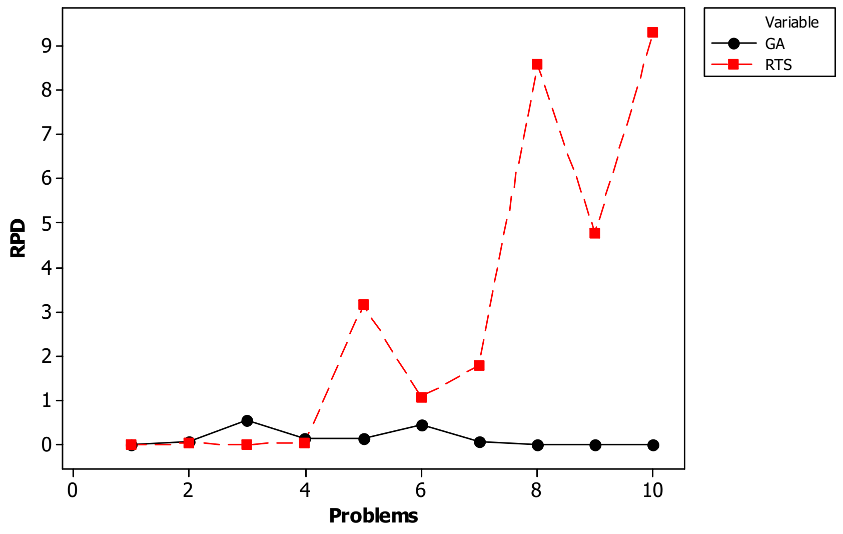

Figure 11. The performance of GA and RTS versus $(p-d)$.

Table 9. The average RPD of the algorithms versus $(v_1 - v_2)$.

$v_1 - v_2$	Genetic	Reactive tabu search
10–10	0.10	2.79
20–20	0.16	2.94
Average	0.13	2.86

Table 10. The average RPD of the algorithms versus m.

M	Genetic	Reactive tabu search
3	0.30	3.30
5	0.03	2.89
10	0.06	2.41
Average	0.13	2.86

that search solution space with one single solution while GA is a population based algorithm and performs the search with a group of solutions.

Table 9 shows the average RPD of the algorithms versus $(v_1 - v_2)$. In both sizes, GA outperforms RTS. Moreover, Figure 12 shows the performance of the proposed algorithms versus (v_1 and v_2). Figure 12 shows that as the number of inbound and outbound trucks increase, the performance of the algorithms becomes worse. GA also keeps its robust performance.

Table 10 shows the average RPD of the algorithms versus m. In all three sizes, GA is again better algorithm than RTS. Figure 13 shows performance of the algorithms versus m. Unlike the previous parameters, by increasing the number of product types, the performance of the tabu search algorithm becomes better.

In order to statistically check the significant difference of the proposed algorithms, one-way analysis of variance test is used. Table 11 shows the results. The results show that there is statistically significant difference between the algorithms with p-value of zero. A total of 240 instances are solved. Thus, the total degree of freedom is 239. The two algorithms are compared. Hence, its degree of freedom is one. The mean square of the algorithm factor is much greater than error. Hence, we can conclude that GA is also statistically better than RTS.

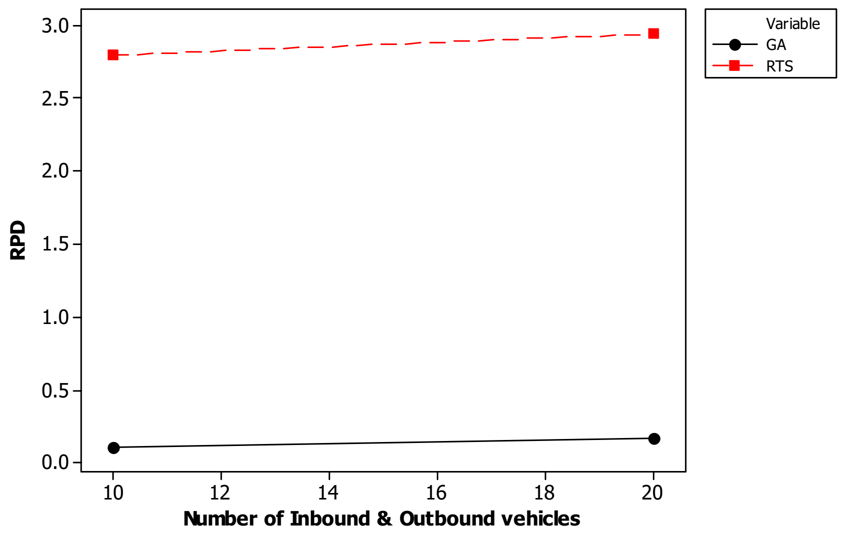

FIGURE 12. The performance of the algorithms versus $(v_1 - v_2)$.

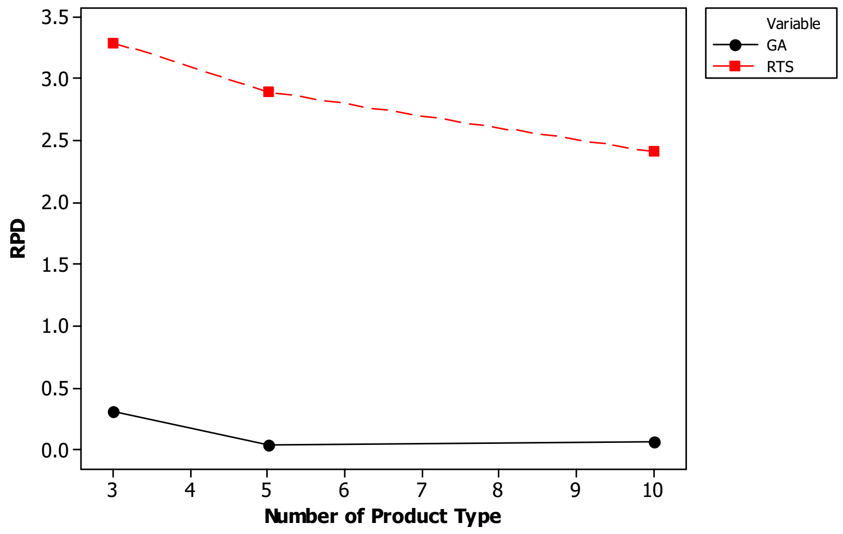

FIGURE 13. The performance of the algorithms versus m.

TABLE 11. Analysis of variance test.

Factor	Degree of freedom	Sum square	Mean square	F_0	P-value
Algorithms	1	449.18	449.18	55.91	0.000
Error	238	1912.12	8.03		
Total	239	2361.30			

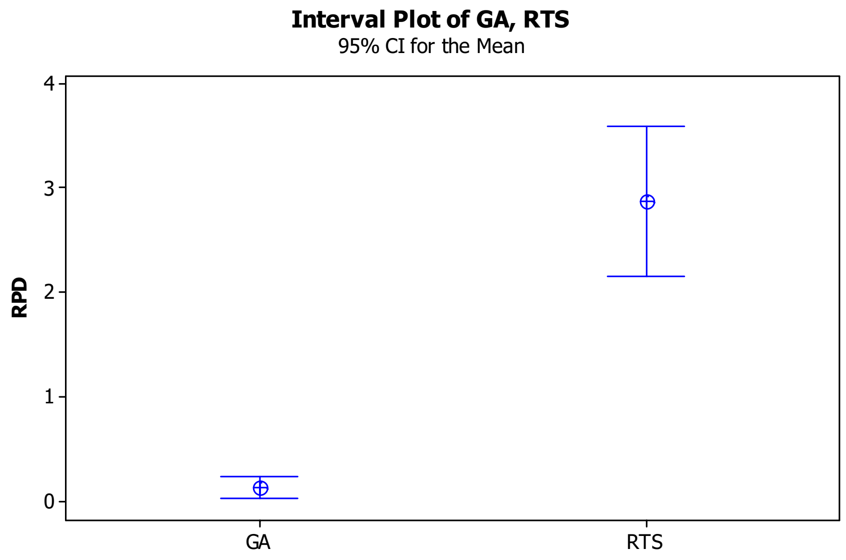

FIGURE 14. The mean and 95% confidence interval for the proposed algorithms.

Figure 14 shows the average RPD and 95% confidence interval for the proposed algorithms. Generally, we can conclude that GA with average RPD value of 0.13% provides better results than RTS with average RPD value of 2.86%.

7. Conclusion and further research

This paper considered the problem of truck routing and scheduling for a cross-dock with one inbound door and one outbound door. The objective was to minimize the makespan with the least number of trucks. Using the specified software of CPLEX, small instances of the problem were solved to optimality. The model is capable of solving instances up to 5 suppliers and 6 costumers. To solve large instances, this procedure was not efficient. Hence, two metaheuristics, based on reactive tabu search and genetic algorithms, were proposed. These algorithms solve the problem efficiently; yet, to evaluate their effectiveness, the algorithms were first tuned and then two experiments were conducted. In the first one, we compare the algorithm for performance against the optimal solution obtained by the mathematical model. In the second one, the performance of the algorithms was further evaluated. On average, the genetic algorithm outperformed the reactive tabu search algorithm.

As an interesting future research, one can extend the problem by considering capacitated trucks. Another further work on this problem is to develop other metaheuristics for this problem. Besides, the problem can be extended by considering a number of inbound and outbound doors for cross-docking system. In addition, we can use a number of cross docks in the distribution network. Also, time windows for suppliers and customers can be given.

References

[1] D. Agustina, C.K.M. Lee and R. Piplani, Vehicle scheduling and routing at a cross docking center for food supply chains. *Int. J. Prod. Econ.* **152** (2014) 29–41.

[2] G. Alpan, R. Larbi and B. Penz, A bounded dynamic programming approach to schedule operations in a cross docking platform. *Comput. Indus. Eng.* **60** (2011) 385–396.

[3] G.A. Alvarez-Perez, J.L. Gonzalez-Velarde and J.W. Fowler, Cross docking – just in time scheduling: an alternative solution approach. *J. Oper. Res. Soc.* **60** (2009) 554–564.

[4] A. Boloori Arabani, S.M.T. Fatemi Ghomi and M. Zandieh, A multi-criteria cross-docking scheduling with just-in-time approach. *Int. J. Adv. Manufact. Technol.* **49** (2010) 741–756.

[5] A. Boloori Arabani, S.M.T. Fatemi Ghomi and M. Zandieh, Metaheuristics implementation for scheduling of trucks in a cross-docking system with temporary storage. *Expert Syst. Appl.* **38** (2011a) 1964–1979.

[6] A. Boloori Arabani, M. Zandieh and S.M.T. Fatemi Ghomi, A cross-docking scheduling problem with sub-population multi-objective algorithms. *Int. J. Adv. Manufact. Technol.* **58** (2011b) 741–761.

[7] A. Boloori Arabani, M. Zandieh and S.M.T. Fatemi Ghomi, Multi-objective genetic-based algorithms for a cross-docking scheduling problem. *Appl. Soft Comput.* **11** (2011c) 4954–4970.

[8] A. Boloori Arabani, F. Ramtin and S.N. Rafienejad, Applying Simulated Annealing Algorithm for Cross-Docking Scheduling, Vol. 2 of *Proc. of the World Congress on Engineering and Computer Science.* San Francisco, USA (2009).

[9] N. Boysen, Truck scheduling at zero-inventory cross docking terminals. *Comput. Oper. Res.* **37** (2010) 32–41.

[10] N. Boysen and M. Fliedner, Cross dock scheduling: classification, literature review and research agenda. *Omega* **38** (2010), 413–422.

[11] F. Chen and K. Song, Minimizing makespan in two-stage hybrid cross docking scheduling problem. *Comput. Oper. Res.* **36** (2009) 2066–2073.

[12] S. Forouharfard and M. Zandieh, An imperialist competitive algorithm to schedule of receiving and shipping trucks in cross-docking systems. *Int. J. Adv. Manufact. Technol.* **51** (2010) 1179–1193.

[13] F. Glover and M. Laguna, Tabu search. Kluwer Academic Publishers, Boston, USA (1997).

[14] F. Glover, Heuristic for Integer Programming Using Surrogate Constraints. *Decis. Sci.* **8** (1977) 156–166.

[15] A.K.R. Jagannathan, *Vehicle routing with cross-docks, split deliveries, and multiple use of vehicles.* Thesis, in Auburn University, Alabama, USA (2011).

[16] Y.H. Lee, J.W. Jung and K.M. Lee, Vehicle routing scheduling for cross-docking in the supply chain. *Comput. Indus. Eng.* **51** (2006) 247–256.

[17] Y. Li, A. Lim and B. Rodrigues, Cross-docking: JIT scheduling with time windows. *J. Oper. Res. Soc.* **55** (2004) 1342–1351.

[18] C.J. Liao, Y. Lin and S.C. Shih, Vehicle routing with cross-docking in the supply chain. *Expert Syst. Appl.* **37** (2010) 6868–6873.

[19] A. Mohtashami, M. Tavana and F.J. Santos-Arteaga, Fallahian-Najafabadi A, A novel multi-objective meta-heuristic model for solving cross-docking scheduling problems. *Appl. Soft Comput.* **31** (2015) 30–47.

[20] F.A. Santos, G.R. Mateus and A.S. Cunha, A Branch-and-price algorithm for a Vehicle Routing Problem with Cross-Docking. *Electron. Notes Discr. Math.* **37** (2011) 249–254.

[21] S. Shahin Moghadam, S.M.T. Fatemi Ghomi and B. Karimi, Vehicle routing scheduling problem with cross docking and split deliveries. *Comput. Chem. Eng.* **69** (2014) 98–107.

[22] R. Soltani and S.J. Sadjadi, Scheduling trucks in cross-docking systems: A robust meta-heuristics approach. *Trans. Res. Part E* **46** (2010a) 650–666.

[23] B. Vahdani and M. Zandieh, Scheduling trucks in cross-docking systems: Robust meta-heuristics. *Comput. Indus. Eng.* **58** (2010) 12–24.

[24] B. Vahdani, R. Soltani and M. Zandieh, Scheduling the truck holdover recurrent dock cross-dock problem using robust meta-heuristics. *Int. J. Adv. Manufact. Technol.* **46** (2010b) 769–783.

[25] B. Vahdani, R. Tavakkoli-Moghaddam, M. Zandieh and J. Razmi, Vehicle routing scheduling using an enhanced hybrid optimization approach. *J. Intel. Manufact.* **23** (2010) 759–774.

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

[27] W. Yu and P.J. Egbelu, Scheduling of inbound and outbound trucks in cross docking systems with temporary storage. *Eur. J. Oper. Res.* **184** (2008) 377–396.