Contents lists available at ScienceDirect

Computers & Industrial Engineering

journal homepage: www.elsevier.com/locate/caie

A new bi-objective vehicle routing-scheduling problem with cross-docking: Mathematical model and algorithms

Asefeh Hasani Goodarzi[a,b], Reza Tavakkoli-Moghaddam[a,*], Alireza Amini[a]

[a] School of Industrial Engineering, College of Engineering, University of Tehran, Tehran, Iran
[b] Département de génie des systèmes, École de technologie supérieure, Montreal, Quebec, Canada

A R T I C L E I N F O

Keywords:
Vehicle routing problem
Cross-docking
Scheduling
Splitting orders
Multi-product
Multi-objective meta-heuristics

A B S T R A C T

This paper addresses a vehicle routing problem with cross-docking (VRPCD) that considers truck scheduling, splitting pickup and delivery orders with time-windows at supplier and retailer locations, while optimizing two conflicting objectives (i.e., cost efficiency and responsiveness). The objectives are to minimize the total operational cost and the sum of the maximum earliness and tardiness. A new bi-objective mixed-integer linear programming model is presented and a multi-objective meta-heuristic evolutionary algorithm is proposed for solving the problem. Numerical results indicate the effectiveness of our proposed algorithm comparing with two multi-objective meta-heuristic algorithms (i.e., non-dominated sorting genetic algorithm (NSGA-II) and Pareto archived evolution strategy (PAES)). Also, we report findings from a hypothetical case study in a retail chain in Houston, Texas.

1. Introduction

Since the beginning of this century, Giant online retailers (e.g., Amazon and Walmart with their transportation fleet) have benefited from cross-docking as a competitive advantage in the market to decrease the cost associated with inventory holding and transportation while providing next- or same-day delivery services to customers. Moreover, third-party logistics (3PL) and carrier companies (e.g., Nexus Logistics and Menlo Logistics) also offer, not exclusively but including, cross-docking services to clients in addition to purchasing, warehousing, transportation, and delivery. In practice, while increasing the delivery cost because of the growing urban population worldwide and surging density of commodity drop in urban areas, demand for home delivery is surprisingly on the rise (Savelsbergh & Woensel, 2016). As such, the need for efficient, responsive, and sustainable delivery services has been a major challenge for not only industrial companies, but also many researchers and experts active in the transport and logistics service industry. Thus, we are motivated to address this research problem, which recognizes the challenge of offering on-time-in-full (OTIF) delivery services to customers in a retail chain sector.

From a business point of view, cross-docking, also known as an efficient strategy to regulate the flow of products in a supply chain, has received global recognition thanks to the advantage of keeping no or at least low level of temporary storage and the economies of scale in inbound and outbound transportation. The main purpose of cross-docking operations in almost all companies in retail chain and logistical services is to collect various supply products in the form of unit pallets, consolidate them into a collection of mixed pallets of outgoing goods with the same destination, and deliver them to retail stores according to the orders.

From a supply chain angle, a company can either utilize its facilities and fleets or outsource those functions to a 3PL service provider. Note that the problem dealt with in this paper can be applied to each of these cases. In the context considered here, the common sequence of events related to cross-docking is given as follows. Heavy-duty trucks collect orders from different suppliers and return to the cross-dock for consolidation. Arriving trucks are then directed to the receiving docks for the pallet unloading process. After consolidation operations, mixed pallets are moved to the staging docks and are then loaded into the outbound trucks. Once it is done, outbound trucks are then dispatched for delivering the pallets (orders) to the retailers in the same or on the next day. Note that incoming pallets are usually delivered to the cross-dock facility in full truckload (FTL), while outgoing pallets are dropped off at delivery points in FTL or less-than-truck-load (LTL) shipments. Goods

* Corresponding author.
E-mail addresses: asefeh.hassani-goodarzi.1@ens.etsmtl.ca (A. Hasani Goodarzi), tavakoli@ut.ac.ir (R. Tavakkoli-Moghaddam), a.amini87@gmail.com (A. Amini).

https://doi.org/10.1016/j.cie.2020.106832
Received 25 August 2019; Received in revised form 31 August 2020; Accepted 5 September 2020
Available online 11 September 2020

consolidations in routing result in the so-called vehicle routing problem with cross-docking (VRPCD), where vehicles start predefined journeys from a cross-dock facility, visit suppliers for picking-up unit pallets in their route and return to the cross-dock for goods consolidation. Fig. 1 illustrates a typical cross-docking network considered in this paper. The solid arcs represent pickup routes of inbound fleets, while the dash-arcs show delivery routes of outbound fleets. It can be seen that all these routes start and end at the cross-dock location and that pickup node 3 and delivery node 4 are met by two vehicles (i.e., their servicing operations have been split).

In the context of the VRPCD, truck scheduling has been considered as a key issue in the pickup, cross-dock, and delivery operations. Lee, Jung, and Lee (2006) first declared that effective physical flow in a chain could be realized by taking pickup, delivery, and cross-docking operations into account. To achieve this goal, vehicle routing and scheduling have to be considered simultaneously to assure smooth physical flow in the chain. The reason behind the latter issue stems from the fact that the late arrival of the inbound fleet to any supplier facility may result in a delay in freight consolidation operations at the cross-dock facility and consequently shortage at retailers in the downstream of a retail chain. On the other hand, the early arrival of a vehicle causes long waiting times at the supplier location that affects the fleet resource utilization (Maknoon & Laporte, 2017). This matter has widely been studied by numerous researchers (see, for example, Ahkamiraad & Wang, 2018; Dondo & Cerdá, 2013; Liao, Lin, & Shih, 2010; Musa, Arnaout, & Jung, 2010; Wen, Larsen, Clausen, Cordeau, & Laporte, 2009).

Both late and early arrivals, which are known as tardiness and earliness, respectively, are due to the lack of timing in freight deployment (Dantzig & Ramser, 1959), which are used to measure customer satisfaction (Lee et al., 2006). Inspired by the concept of just-in-time (JIT) settings to minimize the total late and early arrivals (Liman & Ramaswamy, 1994; Sidney, 1977), we incorporate truck scheduling with OTIF in our study. It is important to mention that both tardiness and earliness on the outbound route give rise to product shortage and truck waiting time, respectively, at retail stores. In general, either case can jeopardize OTIF services in the retail chain sector. In cross-docking network design, vehicle routing decisions for the inbound flow should be taken along with those of the outbound flow, as coordination of inbound and outbound route settings at the cross-dock are strongly interrelated. Therefore, it is important to incorporate the truck scheduling characteristics in routing problems to assure efficient and effective delivery services, (Boysen & Fliedner, 2010; Ladier & Alpan, 2016; Wen et al., 2009).

Overall, this paper proposes an optimization model and a solution algorithm, which tackles a multi-objective VRP with cross-docking, truck scheduling, and order splitting, combined. In this study, we focus on a multi-product flow, which means that each retailer can receive more than one type of goods with different sizes in a truck. To verify our proposed algorithm, the obtained solutions are compared with those of two commonly used algorithms (i.e., the non-dominated sorting genetic algorithm (NSGA-II) and Pareto-archived evolution strategy (PAES)) over a set of instances, which confirm its efficiency.

The remainder of the paper is organized as follows. In Section 2, a brief review of the respective literature to illuminate the background of the problem is presented. The problem description and the formulation of the mathematical model are given in Section 3. The proposed solution algorithm based on a multi-objective meta-heuristic method and benchmarking algorithms is presented in Section 4. Numerical results are reported in Section 5 consisting of a set of instances borrowed from the literature and hypothetical case data. Finally, Section 6 concludes the paper with computational performance, key takeaways about the managerial aspects of this problem, and suggestions for further development.

2. Literature review

In practice, OTIF delivery plays an important role in creating value-added to the retail chain and its stockholders. Cross-docking platforms are considered as a representative of variant sharing economies, which widely has been developed in the related literature. Matzler, Veider, and Kathan (2015) underlined the impact of collaborating logistics service-providers and sharing the available resource and capacity on better consolidation, higher capacity utilization, and reduction in transportation cost. Archetti, Savelsbergh, and Speranza (2008) argued that allowing split deliveries could produce quality distribution operations with better space and time resource utilization. The impact of splitting pickup and delivery operations on better fleet capacity utilization and on improving corresponding service time has been first introduced in Dror and Trudeau (1989) and then developed in Silva, Subramanian, and Ochi (2015), Han and Chu (2016), and Wang, Jagannathan, Zuo, and Murray (2017).

When recognizing splitting pickup and delivery, a retailer's order is fulfilled on multiple routes by different vehicles, which results in a multiple-visit VRPCD. Considering that, Jiang, Ng, Poh, and Teo (2014), Hasani Goodarzi and Zegordi (2016), and Baniamerian, Bashiri, and Tavakkoli-Moghaddam (2019) studied the impact of a heterogeneous fleet on the cross-docking network operational cost and remarkable cost savings using different configurations of the vehicles fleet. Battara, Cordeau, and Iori (2014) and Chen, Chou, Hsueh, and Yen-Ju (2015) also emphasize the paired pickup and delivery with VRP to reduce the transportation cost. Chen, Li, and Liu (2014) studied a multiple-visit feature of an unpaired pickup and delivery problem with multiple commodities, in which every customer is allowed to be visited more than once by different vehicles and is visited at most once by the same vehicle. Then, Xu, Li, Zou, and Liu (2017) extended the work carried out by Chen et al. (2014) and considered a multi-visit concept, in which every customer is allowed to be visited more than once by the same or different vehicle(s). This assumption makes their problem more complicated but more realistic. They showed that the quality of operations was improved and the total transportation cost was reduced.

The importance of truck scheduling has been studied in the literature review, for example in Boloori Arabani, Fatemi Ghomi, and Zandieh, Alpan, Larbi, and Penz (2011); Vahdani, Tavakkoli-Moghaddam, Zandieh, and Razmi (2012); Ghannadpour, Noori, and Tavakkoli-Moghaddam (2014); Mousavi, Tavakkoli-Moghaddam, and Jolai (2013), Amini and Tavakkoli-Moghaddam (2016); Luo, Yang, and Wang (2019); Dulebenets (2019). For the reason explained in the above-mentioned studies, the departure time of outbound vehicles, which are ready to deploy from the cross-dock, depends in large on the arrival time of inbound vehicles forwarded to a set of suppliers to pick the orders and on the service operation time carried out inside the cross-dock (Santos, Mateus, & da Cunha, 2013). Boysen and Fliedner (2010) highlighted the impact of coordination in VRPCD by including truck scheduling for

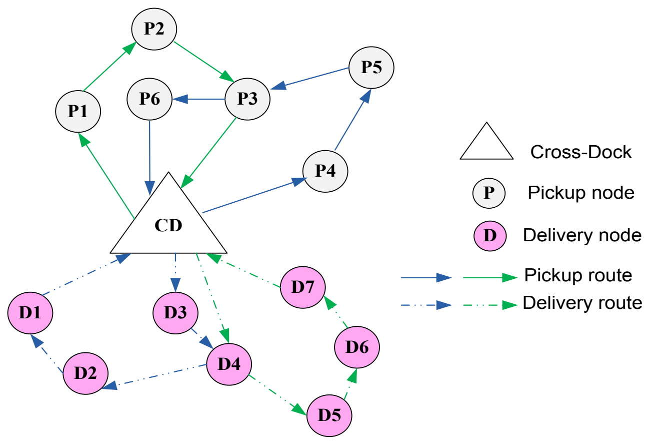


Fig. 1. Cross-docking network with splitting pickup and delivery.

arrival and departure times at each node in the cross-dock transportation network. In a comprehensive review of the pickup and delivery problem, Grimault, Bostel, and Lehuédé (2017) have shown that truck scheduling plays a vital role in maximizing resource unitization and minimizing transportation costs.

In the context of the VRPCD, time windows parameters are usually given as input parameters, which are often the desired pickup and delivery time windows at the supplier and retailers node, respectively. Lehuédé, Péton, and Tang (2014) showed that optimizing time windows for each vehicle on each route significantly improves the customer service level. Ladier and Alpan (2016), Liu, Jiang, Fung, Chen, and Liu (2010), and Neves-Moreira, Amorim, Guimares, and Almada-Lobo (2016) have also stressed the inclusion of vehicle timing in the VRPCD for producing a considerable cost saving for a homogeneous fleet of vehicles. It should be noticed that these values are interrelated to the inbound and outbound transportation times as well as service operation times at the cross-dock. Moreover, the major challenge is to estimate an arrival time within the desired pickup and delivery time windows that are requested by suppliers and retailers, respectively. A vehicle may arrive earlier (later) than the promised time. Thus, to avoid this violation, vehicle scheduling should be jointly considered in vehicle routing optimization (Chen & Lee, 2009; Chen & Song, 2009).

As far as the NP-hardness of the VRPCD and the truck scheduling problem is concerned (Healy & Moll, 1995), it is important to consider an efficient solution method to achieve suitable results. In the relevant literature review, several heuristic algorithms have been developed to identify Pareto solutions and to achieve the best compromise among all objective functions. The comparison of solutions of conflicting objectives can be explained by the state of Pareto optimality and dominance rule that allow solutions to be compared and ranked without applying any measures, such as relative importance in the form of individual weights for objectives (Marseguerra, Zio, & Podofillini, 2002). Over the last few years, several algorithms have been developed to solve multi-objective variants of VRPs, such as the competitive algorithm (Atashpas-Gargari & Lucas, 2007; Javanmard, Vahdani, & Tavakkoli-Moghaddam, 2014), the evolutionary algorithm with simulated annealing (Amini & Tavakkoli-Moghaddam, 2016; Baños, Ortega, Gil, Márquez, & Toro, 2013; Shahmardan & Sajadieh, 2020), scatter search (Melián-Batista, Santiago, Angel-Bello, & Alvarez, 2014), quantum-genetic algorithm (Kourank Beheshti, Hejazi, & Alinaghian, 2015), similarity-based selection (García-Nájera, Bullinaria, & Gutiérrez-Andrade, 2015), bio-inspired optimization algorithms (Hasani Goodarzi & Zegordi, 2016; Kumar et al., 2016; Chen, Hsiao, Reddy, & Tiwari, 2016; Yin & Chuang, 2016; Rabbani, Heidari, Farrokhi-Asl, & Rahimi, 2018; Mollanoori, Tavakkoli-Moghaddam, Triki, Hajiaghaei-Keshteli, & Sabouhi, 2019), and variable neighborhood search (Baniamerian et al., 2019; Hassanzadeh & Rasti-Barzoki, 2017), to name a few. Although monetary and responsiveness objective functions introduced in this paper have a crucial impact on the vehicle routing and deployment decisions, simultaneous optimization of two latter objectives has received little attention in the related literature. In this paper, we develop a multi-objective evolutionary algorithm, which can produce better solutions by combining key features of competing methodologies and embedding local search techniques into evolutionary approaches.

With this motivation, we are interested in solving a problem that involves splitting pickup at supplier nodes and delivery at retailer nodes. More precisely, the model developed in this paper determines the quantity of pickup and delivery of each product at each supplier and retailer location by each vehicle, although total demand and supply at each node are known in advance. Moreover, we consider multi-product settings and a heterogeneous fleet of vehicles in the model formulation. To tackle the truck scheduling issue, we incorporate the earliness/tardiness of a heterogeneous fleet of vehicles carrying multiple products in the proposed model, which ensures accurate and fast flow of a set of products throughout the chain. In this paper, we strive to find the best solution to the operational routing and timing of vehicle deployment while simultaneously optimizing the cost efficiency and customer responsiveness objectives. The former objective refers to transportation and operational costs minimization, while the latter deals with keeping the possible earliness or tardiness at the minimum level. For solving this problem, a multi-objective evolutionary algorithm is developed, which can find better solution fronts comparing to standard multi-objective algorithms. Overall, the incremental contributions of this paper are three-fold:

(i) simultaneous optimization of two conflicting objectives (i.e., costs minimization and earliness/tardiness minimization) in the form of a bi-objective mixed-integer linear programming model,
(ii) integration of truck scheduling (i.e., vehicle earliness/tardiness), and splitting pickup and delivery orders in a multi-product VRPCD, and
(iii) development of a proposed multi-objective meta-heuristic algorithm.

To the best of our knowledge, a problem with the aforementioned characteristics has not been studied in the literature.

3. Problem formulation

3.1. Problem definition

We denote the suppliers set by $\mathscr{P} = \{1, \cdots, P\}$ (i.e., pickup nodes) and the retailers set by $\mathscr{D} = \{1, \cdots, D\}$ (i.e., demand nodes). The cross-dock node is represented by 0. Let us denote the set of all nodes in the network, including pickup nodes, delivery nodes, and the cross-dock node, by $\mathscr{N} = \{1, \cdots, N\}$. We also assume that a node cannot be a pickup and delivery location, simultaneously. Moreover, pickup and delivery processes are not intermixed (i.e., inbound and outbound routes are separated from each other). Vehicles complete the pickup routes and return to the cross-dock before starting the process of delivering.

We define a desired arrival time windows $[e_i, l_i]$, where e_i and l_i represent the earliest and the latest desired arrival time, respectively, at node $i \in \mathscr{N}\backslash\{0\}$. However, each vehicle is allowed to arrive at node $i \in \mathscr{N}\backslash\{0\}$ no earlier than E_i and no later than L_i. In other words, the desired time windows $[e_i, l_i]$, in which an individual node requests to be visited, and the allowable time windows $[E_i, L_i]$, in which a vehicle is allowed to visit the corresponding node, are taken into account when deciding for inbound and outbound routes. It should be clear that interval $[E_i, L_i]$ includes $[e_i, l_i]$. Any arrival time violation within $[E_i, L_i]$ but not in $[e_i, l_i]$ results in earliness/tardiness and reduces customer satisfaction and service level. This timeline is illustrated in Fig. 2.

The pickup and delivery operations are carried out via a set of vehicles $\mathscr{V} = \{1, \cdots, V\}$, which are assumed heterogeneous. A heterogeneous fleet of vehicles relaxes the same capacity assumption and allows the model to get different fleet sizes for the inbound and outbound vehicles. For this purpose, the capacity of vehicle $m \in \mathscr{V}$ is denoted by Q_m. As previously mentioned, it is assumed that vehicles cannot pickup/deliver at the same node simultaneously and that vehicles for pickup orders are different from those that are used for order delivery. Other assumptions made in this modelling approach are given below.

The set of product families are defined by $\mathscr{R} = \{1, \cdots, R\}$. We denote demand of retailer at node $i \in \mathscr{D}$ for product family $r \in \mathscr{R}$ by d_{ir} and the supply amount of product family $r \in \mathscr{R}$ available at supplier node $i \in \mathscr{P}$ by b_{ir}. As discussed before, splitting pickup and delivery allows vehicles to visit a typical supplier (retailer) several times on different routes. See Fig. 1 for a better understanding. In this regard, each node contains at least two paths, outgoing and incoming paths. A path includes unique origin and destination nodes, in which a vehicle should be assigned to each path. It should be noticed that although demand or supply of some nodes may be split, they are satisfied eventually. The overall demand and supply equilibrium conditions for each product family $r \in \mathscr{R}$ have to be met in the network under investigation (i.e., $\sum_{i \in \mathscr{D}} b_{ir} = \sum_{i \in \mathscr{P}} d_{ir}$). It is

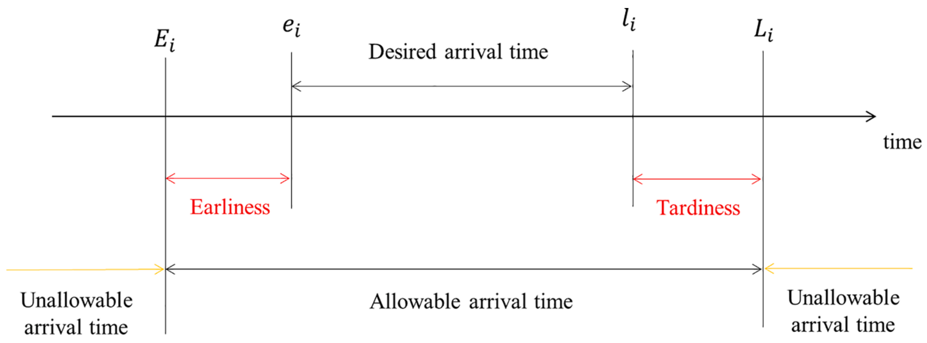


Fig. 2. Vehicle arrival timeline at a node.

worth mentioning that the quantity of demand to deliver at a retailer node and that of supply to pick up at a supplier node by each vehicle are decision variables and are optimized in the proposed modeling framework.

We denote the transportation cost from node i to node j is denoted by tc_{ij}. It should be noticed that $tc_{ij} = \tau.D_{ij}$, where τ and D_{ij} refer to the transportation cost per distance unit and the distance from node i to node j, respectively. Service operational cost, indicated by oc_m, is the cost associated with loading/unloading and material handling operations, which takes place at the retailer/supplier location. This operational cost is accounted for each vehicle leaving the cross-dock for servicing node j. Recall that node j can be of any nodes in the set of suppliers or retailers. We introduce the decision variables used in the model as follows:

- x_{ij}^m is the binary variable that returns one if vehicle m moves from node i to node j, and zero otherwise, x_{0j}^m is the binary variable that returns one if vehicle m moves from the cross-dock (i.e., node 0 to node j), and zero otherwise,
- y_i^m is a binary variable that returns one if vehicle m visits node i, and zero otherwise.
- Er_i^m denotes the earliness of vehicle m at node i,
- Tr_i^m denotes the tardiness of vehicle m at node i,
- a_{ir}^m denotes a non-negative continuous decision variable, which determines the amount of supply of product r to be picked up by vehicle m at supplier node i.
- z_{ir}^m denotes a non-negative continuous decision variable, which determines the amount of demand of product r to be delivered by vehicle m at retailer node i

3.2. Problem modeling

In this section, we propose a mixed-integer linear programming (MILP) model, which optimizes the cost efficiency and responsiveness criteria simultaneously, considering the structural constraints of routing, order splitting with multiple-visit, and capacitated fleet size. 3PLs and carrier provider companies depending upon their competitive strategy determine how the distribution operations should be performed concerning the efficiency and responsiveness objectives. For example, companies such as Walmart and Amazon, which utilize their transportation fleet and cross-docking facilities, rely not only on efficiency to achieve minimum operational cost, but also on responsiveness and product availability to maintain customer satisfaction level as high as possible.

3.2.1. Objective functions

The optimization model presented in this section aims to minimize two conflicting objectives (i.e., cost efficiency and responsiveness criteria) presented in the objective functions (1) and (2), respectively. The former refers to the monetary objective function that minimizes the total operational cost, as follows:

$$\text{Min} \sum_{i\in\mathscr{N}}\sum_{j\in\mathscr{N}}\sum_{m\in\mathscr{V}} tc_{ij}x_{ij}^m + \sum_{j\in\mathscr{N}\backslash\{0\}}\sum_{m\in\mathscr{V}} oc_m x_{0j}^m \tag{1}$$

The second objective reflects on the level of responsiveness that corresponds to the vehicle deployment scheduling so that it minimizes the possible earliness and tardiness at pickup and delivery nodes, given in the objective function (2). The amount of time when the vehicle m is in earliness and tardiness is described as the length of time violated from the desired arrival time windows at node j, as follows:

$$\text{Min} \sum_{i\in\mathscr{N}\backslash\{0\}}\sum_{m\in\mathscr{V}} \left(Er_i^m + Tr_i^m\right) \tag{2}$$

This objective aims to minimize the earliness and tardiness of all fleet vehicles operating in the network. The system constraints are presented below.

3.2.2. Vehicle scheduling constraints

We denote the actual arrival time of vehicle m to node i by p_i^m. As depicted in Fig. 2, an earliness happens when a vehicle arrives earlier than the desired time (i.e., $e_i > p_i^m$) and tardiness happens when a vehicle arrives later than the desired time (i.e., $p_i^m > l_i$). Earliness and tardiness are defined as the violation of the actual arrival time from the promised time and presented as follows: $Er_i^m = max\{0, e_iy_i^m - p_i^m\}$ and $Tr_i^m = max\{0, p_i^m - l_iy_i^m\}$, respectively. linear earliness and tardiness functions are given in Constraints (3) and (4), respectively.

$$Er_i^m \geq e_iy_i^m - p_i^m \quad i\in\mathscr{N}\backslash\{0\}, m\in\mathscr{V} \tag{3}$$

$$Tr_i^m \geq p_i^m - l_iy_i^m \quad i\in\mathscr{N}\backslash\{0\}, m\in\mathscr{V} \tag{4}$$

It should be clear that if the arrival time takes place within the desired time windows $[e_i,l_i]$, the objective function (2) will be equal to 0. However, each vehicle is allowed to reach the assigned node within the given allowable time windows $[E_i, L_i]$ as presented by:

$$E_iy_i^m \leq p_i^m \leq L_iy_i^m; \quad i\in\mathscr{N}\backslash\{0\}, m\in\mathscr{V} \tag{5}$$

3.2.3. Vehicle synchronization constraints

Constraint (6) satisfies the timing structure, in which vehicle m moves from node i at time p_i^m for visiting node j on its route at time p_j^m, as given by:

$$p_i^m + Ser_i + t_{ij} - M\left(1 - x_{ij}^m\right) \leq p_j^m; \quad i\in\mathscr{N}, j\in\mathscr{N}, m\in\mathscr{V} \tag{6}$$

where the required time for pickup or delivery operations spent at node i is given by Ser_i and the required time to travel from node i to node j is denoted by t_{ij}. In this constraint, M denotes a large positive number. Constraint (6) also guarantees sub-tours elimination. It prevents backward movement in routes and guarantees the continuity of routes.

Total operational time spent at node i and total transportation time for traveling from node i to node j in the network are considered when planning for vehicles deployment over the time horizon T, as given by:

$$\sum_{i\in\mathscr{N}}\sum_{j\in\mathscr{N}}(Ser_j+t_{ij})x_{ij}^m \leq T; \quad m\in\mathscr{V} \tag{7}$$

As can be seen, vehicle synchronization constraints, which correspond to the responsiveness objective (earliness and tardiness) of a 3PL or a carrier provider company, are coupled with the routing decisions. This set of constraints confirms the strong interrelationship between vehicle routing and cross-docking operations.

3.2.4. *Splitting pickup and delivery constraints*

Constraint (8) allows the order quantity at supplier $i\in\mathscr{P}$ for product family $r\in\mathscr{R}$, denoted by b_{ir}, can be partially or completely picked-up by vehicle m and Constraint (9) assures that the total amount of supply orders that has been split into several vehicles ultimately satisfies total supply amount of product r available at supplier node i.

$$a_{ir}^m \leq b_{ir}y_i^m; \quad i\in\mathscr{P}, r\in\mathscr{R}, m\in\mathscr{V} \tag{8}$$

$$\sum_{m\in\mathscr{V}}a_{ir}^m = b_{ir}; \quad i\in\mathscr{P}, r\in\mathscr{R} \tag{9}$$

Similarly, Constraint (10) verifies that the demand of retailer $i\in\mathscr{D}$ for product family $r\in\mathscr{R}$, denoted by d_{ir}, can be delivered partially by several vehicles or completely by a single vehicle. Constraint (11) guarantees that the total demand of retailer node i is met at the end of the day.

$$z_{ir}^m \leq d_{ir}y_i^m; \quad i\in\mathscr{D}, r\in\mathscr{R}, m\in\mathscr{V} \tag{10}$$

$$\sum_{m\in\mathscr{V}}z_{ir}^m = d_{ir}; \quad i\in\mathscr{D}, r\in\mathscr{R} \tag{11}$$

Overall, Constraints (8)–(11) highlight key features of splitting pickup and delivery of multiple products carried by the vehicles within the network to achieve the economies of scale. These constraints also specify the optimal amount of each product to be picked up at supplier nodes and to be dropped at retailer nodes.

3.2.5. *Fleet capacity and load constraints*

From a commercial point of view, a 3PL or a transportation provider company needs to determine the size and the composition of its shipping fleets. Heterogeneous fleets allow a company to specify a set of required vehicles with various capacities as well as transportation and operational costs. These characteristics are presented in Constraints (12)–(14). Constraint (12) ensures that the number of each type of vehicle leaving the cross-dock cannot exceed the maximum number of vehicles in the network.

$$\sum_{j\in\mathscr{N}}x_{0j}^m \leq V^m; \quad m\in\mathscr{V} \tag{12}$$

where V^m denotes the maximum number of available vehicles of type m in the network. Constraint (13) ensures that the accumulated volume of products loaded at pickup (supplier) node must not exceed the vehicle's capacity. The vehicle capacity constraint also holds for outbound fleets so that the accumulated volume of products loaded into the trucks at cross-dock must not exceed the outbound vehicle capacity.

$$\sum_{i\in\mathbb{P}}\sum_{r\in\mathscr{R}}f_r a_{ir}^m \leq Q_m; \quad m\in\mathscr{V} \tag{13}$$

$$\sum_{i\in\mathbb{D}}\sum_{r\in\mathscr{R}}f_r z_{ir}^m \leq Q_m; \quad m\in\mathscr{V} \tag{14}$$

where Q_m and f_r denote the load capacity of vehicle m in volume and the volume of product type r, respectively.

3.2.6. *Cross-dock vehicle routing constraints*

As discussed before, a cross-dock vehicle routing problem consists of a set of paths that originates from the cross-dock, passes through a certain set of nodes, and terminates at the cross-dock. These constraints are presented in Constraints (15)–(19). Constraints (15) and (16) guarantee that each vehicle starts its journey from a cross-dock for visiting a set of nodes and returns to the cross-dock at the end of the journey. Recall that the cross-dock node is represented by node 0.

$$\sum_{j\in\mathscr{N}}x_{0j}^m \leq 1; \quad m\in\mathscr{V} \tag{15}$$

$$\sum_{j\in\mathscr{N}}x_{j0}^m \leq 1; \quad m\in\mathscr{V} \tag{16}$$

Constraint (17) ensures that a deployed vehicle has to arrive at a supplier node and moves forward to the next supplier node. This constraint also holds for the retailer nodes, represented in Constraint (18). Once the problem solved, one can determine which vehicles visit a typical supplier or retailer node using these two constraints.

$$\sum_{i\in\mathscr{P}\cup 0}x_{ij}^m = y_j^m; \quad j\in\mathscr{P}, m\in\mathscr{V} \tag{17}$$

$$\sum_{i\in\mathscr{D}\cup 0}x_{ij}^m = y_j^m; \quad j\in\mathscr{D}, m\in\mathscr{V} \tag{18}$$

The sequential movement of vehicles is guaranteed in Constraint (19), which implies that when a vehicle enters a node, it must exit it eventually.

$$\sum_{i\in\mathscr{N}}x_{ik}^m = \sum_{j\in\mathscr{N}}x_{kj}^m; \quad k\in\mathscr{N}\backslash\{0\}, m\in\mathscr{V} \tag{19}$$

3.2.7. *Multiple-visiting constraints*

Contrary to the traditional VRP that forces a vehicle to visit nodes only once, splitting pickup and delivery settings allow vehicles to visit nodes more than once which are represented in Constraints (20)–(22). Constraint (20) demonstrates the possibility of splitting orders in pickup and delivery processes. This constraint guarantees that a node can be visited more than once by multiple vehicles, which are destined from different nodes in the network. Constraints (21) and (22) prevent intermixing the pickup and delivery processes.

$$\sum_{m\in\mathscr{V}}\sum_{i\in\mathscr{N}}x_{ij}^m \geq 1; \quad j\in\mathscr{N}\backslash\{0\} \tag{20}$$

$$\sum_{m\in\mathscr{V}}\sum_{j\in\mathscr{P}}x_{ij}^m = 0; \quad i\in\mathscr{D} \tag{21}$$

$$\sum_{m\in\mathscr{V}}\sum_{j\in\mathscr{D}}x_{ij}^m = 0; \quad i\in\mathscr{P} \tag{22}$$

3.2.8. *Variable domains*

The binary and the non-negative continuous decision variables involved in the problem are given in Expressions (23) and (24), respectively.

$$y_i^m, x_{ij}^m \in\{0,1\}; \quad i\in\mathscr{N}, j\in\mathscr{N}, m\in\mathscr{V} \tag{23}$$

$$a_{ir}^m, z_{ir}^m, Tr_j^m, Er_j^m, p_i^m \geq 0; \quad i\in\mathscr{N}, j\in\mathscr{N}, m\in\mathscr{V}, r\in\mathscr{R} \tag{24}$$

The presented model contains $NV(N+R+5)+NR+2N+6V$ linear constraints, $NV(N+1)$ binary decision variables, and $NV(R+1)$ non-negative continuous decision variables. We can see that, the traditional dial-a-ride problem can be a particular case of the problem under investigation in this paper if the multiple products of cross-docking and splitting order features are relaxed. As the NP-hardness of the traditional dial-a-ride problem has been widely known (Healy & Moll, 1995), the MILP model in this paper is proven as the NP-hard one.

4. Solution methods

It should be clear that the VRPCD is classified as an NP-hard problem and thus an efficient solution algorithm to find the optimal or near-optimal solutions in reasonable computation time is required. Unfortunately, the commercial optimization solvers are unable to converge to optimality in a short time when the instance size increases. In the following, a new multi-objective evolutionary algorithm is constructed based on key features of competing methodologies and embedding local search techniques into evolutionary approaches. It has been illustrated that our proposed solution methodology produces more efficient Pareto fronts.

4.1. *Multi-objective evolutionary algorithm*

In this section, we aim to obtain non-dominated solutions for the problem under investigation in this paper. We develop a proposed multi-objective evolutionary algorithm (MOEA), in which the non-dominant solution generation technique, crowding distance calculation, and evolutionary and local search strategies are used to generate efficient Pareto solutions. Inspiring from the imperialist competitive algorithm (ICA) by the framework proposed by Atashpas-Gargari and Lucas (2007), we develop a multi-objective evolutionary algorithm incorporating efficient local search strategies that allow the algorithm to converge in a faster pace. The reason we have chosen this framework is that its characteristics are easily adaptable with the formulated mixed-integer programming.

4.1.1. *Initial solution representation*

Initial solutions are presented in the form of a matrix. Let us define the fraction of the product that is carried by a vehicle when visiting pickup and delivery nodes by $E^{pickup}_{V\times N}$ and $E^{delivery}_{V\times N}$, respectively, where V denotes the total number of available vehicles and N is the number of nodes in the network. The element of each matrix (i.e., $e^{p}_{mi} \in E^{pickup}_{V\times N}$ and $e^{d}_{mi} \in E^{delivery}_{V\times N}$) represents the fraction of pickup and delivery amount, respectively, at node $i \in \mathscr{N}$ by vehicle $m \in \mathscr{V}$, percentage-wise. The initial values of e^{p}_{mi} and e^{d}_{mi} are generated randomly within the interval $[0\%, 100\%]$ such that the percentage of picked-up (delivered) values carried by all vehicles should satisfy the total demand (supply) of that node. An illustrative example of $E^{delivery}_{V\times N}$ with $V = 5$ and $N = 8$ is given in Fig. 3, in which each element in the matrix corresponds to the percentage of demand delivered at node i (in the column) by vehicle m (in the row). It can be seen that the first column of the matrix states that vehicles 1, 2, 4, and 5 drop 10%, 30%, 51%, and 9%, respectively, of a particular product at delivery node 1 and that vehicle 3 does not visit node 1. It is trivial that the percentage over each column of the matrix should accumulate to 100% meaning that the total amount delivered to the respective node is satisfied.

Besides, we define the routing solutions in the form of a matrix as follows. As mentioned before, the pickup and delivery processes are not intermixed, and thus, the routing matrices of pickup and delivery processes are defined separately as follows, $F^{pickup}_{V\times N} = [f^{p}_{mi}]$ and $F^{delivery}_{V\times N} = [f^{d}_{mi}]$, respectively, where V is the total number of available vehicles and N is the number of nodes. The elements of each matrix, i.e., f^{p}_{mi} and f^{d}_{mi}, represent the sequence of supply and demand nodes visited on picking-up and delivering routes by vehicle m, respectively. For the pickup and delivery routing matrices given by $F^{pickup}_{V\times N}$ and $F^{delivery}_{V\times N}$, the cross-dock is considered as the first and last nodes. Therefore, every tour should begin and end with it. To do so, the cross-dock is added to all constructed routes to calculate the time and the cost associated with each route. To construct a route, the initial sequence of nodes visited by a vehicle is randomly generated. An illustrative example of $F^{delivery}_{V\times N}$ with $V = 5$ and $N = 8$ is shown in Fig. 4, in which each element represents the order of visiting node i, in a column, that is visited by vehicle m in a row. For example, the sequence of nodes visited by vehicle 1 leaving the cross-dock starts from node 2 and continues with nodes 1, 3, 8, and finally 5, respectively. This vehicle returns to the cross-dock at the end of the route. If all elements in an array are set to zero, the corresponding vehicle is not used in the process.

4.1.2. *Solution evaluation measure*

We apply the non-dominance sorting criterion (Goldberg, 1989) technique for measuring the quality of individual solutions generated by the algorithm for ranking and selection. This criterion is measured by calculating the crowding distance measure and the solution fitness criterion (Deb, Pratap, Agarwal, & Meyarivan, 2002) for selecting the good Pareto solutions, based on which, we rank the individual generated Pareto solutions in the multi-objective structure, as explained in the following.

<u>Non-dominance Pareto optimality technique</u>

Assume objective functions $f_{\mathscr{m}}(x); \mathscr{m} \in \mathscr{M} = \{1, 2, \cdots, |\mathscr{M}|\}$, to be optimized in a single model, where x represents a set of decision variables of that model. The dominance rule used in this algorithm is given in the following proposition:

(a) Solution set x_1 dominates solution set x_2, denoted by $x_1 \succ x_2$, if
(i) $f_{\mathscr{m}}(x_1) \le f_{\mathscr{m}}(x_2), \forall \mathscr{m} \in \mathscr{M}$. It implies solution x_1 is not worse than solution x_2 for all objective functions.
(ii) $f_{\mathscr{m}}(x_1) < f_{\mathscr{m}}(x_2),\ \exists \mathscr{m} \in \mathscr{M}$. It implies there exists at least one objective function of which solution x_1 is better than solution x_2.
(b) Solution sets x_1 and x_2 do not dominate each other, denoted by $x_1 \not\approx x_2$, if $f_{\mathscr{m}}(x_1) = f_{\mathscr{m}}(x_2), \forall \mathscr{m} \in \mathscr{M}$. It implies solution sets x_1 and x_2 are placed in the same front.

Note that non-dominated solution sets (i.e., solutions that are not dominated by other solutions) are denoted by x^{*} and are positioned in the first Pareto front solutions (i.e., $x^{*} \in \mathscr{F} = 1$). Accordingly, the solution set in the second front is composed of solutions that are only dominated by solutions of the first front, and so on.

<u>*Crowding distance*</u>: The crowding distance ($cd_{\mathscr{r}}$) used in the proposed multi-objective evolutionary algorithm is a measure that evaluates the quality of two neighboring solutions of a Pareto front. In other words, this measure represents the density of solutions surrounding a particular solution. Let us assume that Pareto front $\mathscr{f}$ is composed of $\mathscr{n}$ individual

Vehicle (m)	Delivery node (i)							
	1	2	3	4	5	6	7	8
1	10%	3%	22%	0%	44%	0%	0%	24%
2	30%	15%	63%	0%	31%	40%	88%	0%
3	0%	21%	0%	0%	0%	60%	0%	0%
4	51%	0%	15%	31%	0%	0%	0%	76%
5	9%	61%	0%	69%	25%	0%	12%	0%
Total	100%	100%	100%	100%	100%	100%	100%	100%

Fig. 3. Solution representation for the delivery amount per vehicle.

	Nodes (i)							
Vehicle (m)	1	2	3	4	5	6	7	8
1	2	1	3	0	5	0	0	4
2	4	1	2	0	6	5	3	0
3	0	1	0	0	0	2	0	0
4	2	0	4	1	0	0	0	3
5	3	5	0	1	2	0	4	0

Fig. 4. Solution representation for a sequence of visiting nodes by vehicles.

solutions (i.e., $x_1, x_2, \cdots, x_n$) and that $f^{\wp}_m(x_i)$ represents the value of m-th objective function of the set of solution x_i included in the Pareto front $\wp$. In the first step, we set the crowding distance associated with each solution, which is denoted by cd_i, equal to zero. In Step 2, we calculate the maximum and minimum values of the m-th objective function in the Pareto front $\wp$ for solution x_i is given by $\overline{f}^{\wp}_m = \max_i\{f^{\wp}_m(x_i)\}$ and $\underline{f}^{\wp}_m = \min_i\{f^{\wp}_m(x_i)\}$, respectively. In Step 3, we define $\wp.m$ as the Pareto front $\wp$, in which individual solutions are sorted in increasing order based on their value for the m-th objective function. Then, we compute the crowding distance of the sorted solutions for the m-th objective function as follows $cd_i = \sum_{m=1}^{\mathscr{M}} \frac{f^{\wp}_m(x_{i+1}) - f^{\wp}_m(x_{i-1})}{\overline{f}^{\wp}_m - \underline{f}^{\wp}_m}$. The crowding distance is a measure to evaluate the diversity of generated solutions, and high crowding distance is preferred to a lower one. Note that solutions on the boundary are assigned an infinite distance value. For further information on the crowding distance calculation, the reader may refer to Deb et al. (2002). Step 4 determines solutions with a higher value of crowding distance which are preferred over other solutions. Therefore, solutions with the highest crowding distance criterion representing the non-dominant solutions are saved to archive. The crowding distance procedure, which represents the density of solutions surrounding a particular one, is shown in Algorithm 1.

Algorithm 1. (*Crowding distance procedure*)

```
INPUT: $n = |\wp|$, set of $|\mathscr{M}|$ objectives.
Step1:    for $i = 1 \cdots n$ do
              $cd_i = 0$ end for
Step2:    for $m = 1, \cdots, |\mathscr{M}|$ do
              $\overline{f}^{\wp}_m = \max_i\{f^{\wp}_m(\mathbf{x}_i)\}$   $\underline{f}^{\wp}_m = \min_i\{f^{\wp}_m(\mathbf{x}_i)\}$ end for
Step3:    for $m = 1, \cdots, |\mathscr{M}|$ do
              SORT $\wp.m$
              $cd_1 = cd_n = \infty$   for $i = 2 \cdots n-1$ do
                  $cd_i = \sum_{m=1}^{\mathscr{M}} \frac{f^{\wp}_m(\mathbf{x}_{i+1}) - f^{\wp}_m(\mathbf{x}_{i-1})}{\overline{f}^{\wp}_m - \underline{f}^{\wp}_m}$   end for
          end for
Step4:    for $i = 1 \cdots n$ do
              $i^* = \mathrm{argmax}\{cd_i\}$ end for
OUTPUT: solution index $i^*$ with the highest crowding distance.
```

Solution fitness: The fitness function is a mechanism to evaluate the goodness of a solution while making a comparison between different possible solutions in a Pareto front using the crowding distance procedure. This function calculates the value of $|\mathscr{M}|$ objective functions associated with n individual solutions in the Pareto front $\wp$. The best solutions which have been found from the solution pool are shortlisted and are denoted by $x_{\mathfrak{n}}$, where $\mathfrak{n} \in \mathfrak{R} = \{1, \cdots, |\mathfrak{R}|\}$ and $\mathfrak{R}$ is the set of dominant solutions. The rest of the individual solutions, which are not included in $\mathfrak{R}$, remain in the solution pool. This latter set is represented by $\mathfrak{C} = \{1, \cdots, |\mathfrak{C}|\}$. In this step, the quality of each dominant solution $x_{\mathfrak{n}}, \mathfrak{n} \in \mathfrak{R}$ is compared with all solutions in $\mathfrak{R}$ considering the crowding distance to the individual solutions in $\mathfrak{R}$. We compute the quality of fitness function of each dominant solution $x_{\mathfrak{n}}, \mathfrak{n} \in \mathfrak{R}$ with other dominant solutions in $\mathfrak{R}$ and denote it by $\mathfrak{p}_{\mathfrak{n}} = cd_i / \sum_{i=1}^{|\mathfrak{R}|} cd_i$. In other words, this latter represents the proportional power of each dominant solution. In this algorithm, we define the number of individual solutions that are dominated by a dominant solution $x_{\mathfrak{n}}, \mathfrak{n} \in \mathfrak{R}$ by $NC = \mathrm{int}\{\mathfrak{p}_{\mathfrak{n}}.|\mathfrak{C}|\} + 1$. This latter number varies over iterations and tends to increase for each dominant solution, as it keeps outperforming throughout the algorithm. It is clear that when the number of solutions that are dominated by a dominant solution increases, the fitness value of the dominant solution increases, and thus the number of dominant solutions in $\mathfrak{R}$ reduces (Atashpas-Gargari & Lucas, 2007).

4.1.3. Evolutionary policies

In this section, we represent evolutionary policies used in the proposed multi-objective evolutionary algorithm for solving our problem. The evolutionary policies applied in this paper are as follows: crossover policy, absorption policy, exchanging policy, and revolution policy. Results have shown that applying these policies allows the algorithm to perform local search more efficiently and to converge to the optimal/near-optimal solutions in a faster way.

Crossover policy: A crossover policy is used for sharing information about the solution pool and improving the quality of dominant solutions. Thus, better solutions have more chances to be selected for sharing the information they carry. One-point and two-point crossover operators for a typical solution of demand percentage delivered by vehicles to demand nodes are shown in Fig. 5(a) and (b), respectively.

Absorption policy: An absorption policy controls random-based absorption of individual solutions towards the best solutions (dominant solutions). The gradual absorption of the set of individual solutions into a dominant solution results in high-quality dominant solutions. This policy integrates individual solutions $x_1, x_2, \cdots, x_n$ with dominant solutions $x_{\mathfrak{n}}, \mathfrak{n} \in \mathfrak{R}$, as explained below.

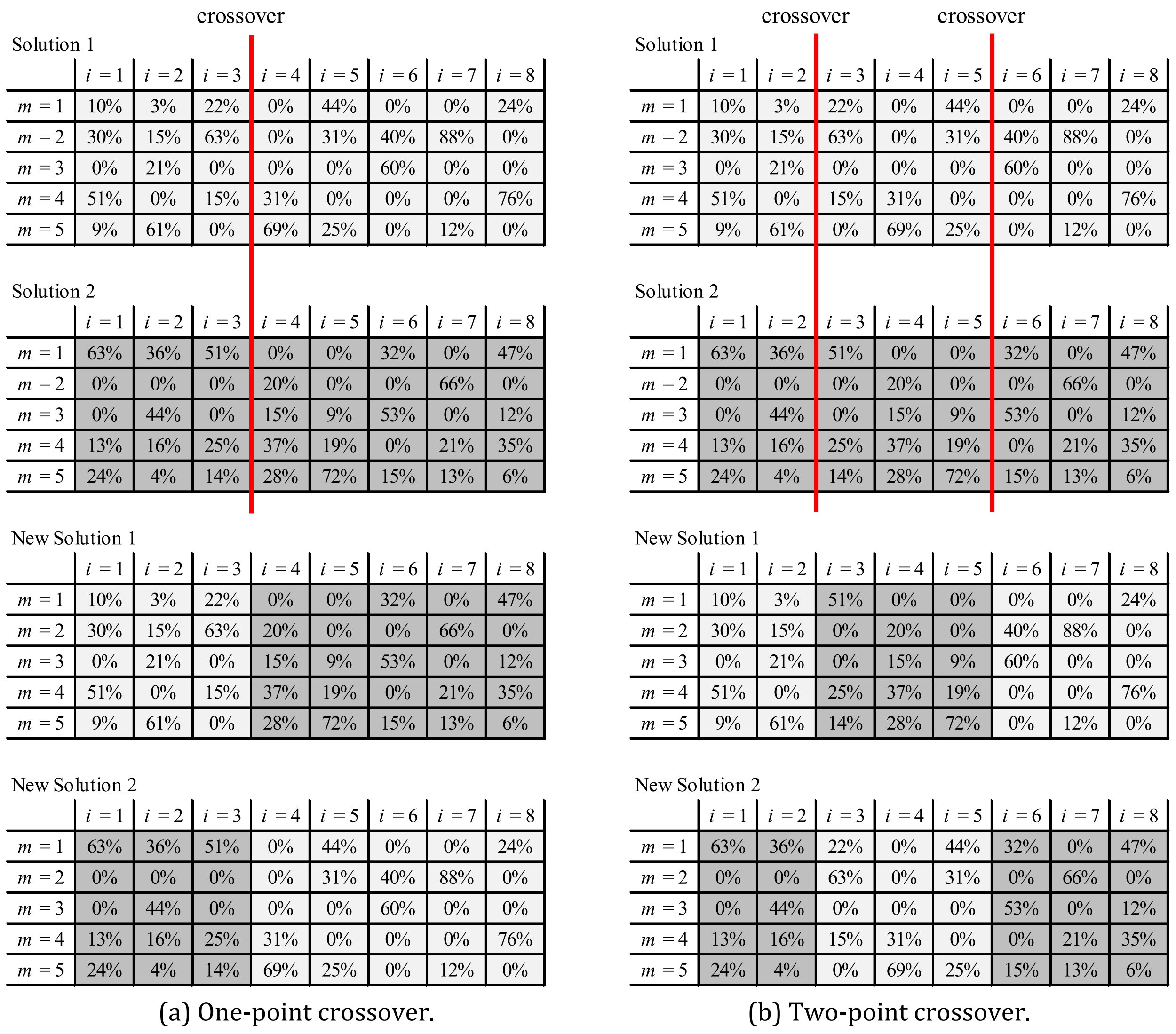
(i) *Distance between a dominant solution and a dominated solution.* This parameter, denoted by d in Fig. 6, represents the distance of dominant-dominated pairs and tends to decrease as the algorithm converges to the stopping criterion. Regarding this parameter, the linear distance between the new and current positions of a dominated solution (α) is decided. The details are mentioned in (iii) *Advancement.*

(ii) *Direction of movement.* It specifies an angle under which an individual solution is directed towards a dominant solution, which is denoted by θ. This direction is randomly generated in a given interval, in degree scale, by $\theta \in [-\gamma, \gamma]$, where γ represents a deviation from the original direction and is set to an arbitrary value. The bigger γ results in a wider search space, while the smaller values for γ creat solutions which are closer to original direction. θ is measured in radians and mostly set to $\pi/4$.

(iii) *Advancement.* It describes the progress of a set of solution $x_1, x_2, \cdots, x_n$ toward reaching better solutions $x_{\mathfrak{n}}, \mathfrak{n} \in \mathfrak{R}$, depending on the evaluation function of newly generated solutions. Based on this parameter, a new solution may approach a better solution by getting better evaluation function and may even dominate a better one. We introduce the advancement value by a distance-dependent random number and denote it by $\alpha \in [0, \Delta d]$., where Δ is a number greater than 1 and d is the distance between two solution points. This value explicitly denotes how close an individual solution is to a dominant solution.

Three functions described above allow the algorithm to perform a more efficient local search. This policy with the associated functions is illustrated in Fig. 6. In this figure, three dominated solutions approach two dominant solutions, thus, there are three dominant-dominated pairs. The linear distance of the new and previous positions is determined by α and the angle of this movement is shown by θ.

Exchanging policy: While moving toward a dominant solution, an individual solution may outperform the corresponding dominant solution, so that such individual solution becomes a dominant solution and the algorithm continues along with the new dominant solution set. For each iteration, we compare every dominant solution to the best individual solutions in terms of the crowding distance in the Pareto front one. The first Pareto front is then sorted in terms of crowding distance, and thus solutions with the highest crowding distance value are selected as a dominant solution. The exchange policy function is illustrated in Fig. 7, in which each bullet demonstrates an individual solution involved in the algorithm and the size of each bullet represents the evaluation function associated with that solution. In this figure, a set of dominated solutions is shown by grey small bullets, while the dominant solutions are presented by black bullets, which are bigger in size for a given


crossover
crossover
crossover
(a) One-point crossover.
(b) Two-point crossover.

(a) One-point crossover.

Solution 1

	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$	$i=6$	$i=7$	$i=8$
$m=1$	10%	3%	22%	0%	44%	0%	0%	24%
$m=2$	30%	15%	63%	0%	31%	40%	88%	0%
$m=3$	0%	21%	0%	0%	0%	60%	0%	0%
$m=4$	51%	0%	15%	31%	0%	0%	0%	76%
$m=5$	9%	61%	0%	69%	25%	0%	12%	0%

Solution 2

	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$	$i=6$	$i=7$	$i=8$
$m=1$	63%	36%	51%	0%	0%	32%	0%	47%
$m=2$	0%	0%	0%	20%	0%	0%	66%	0%
$m=3$	0%	44%	0%	15%	9%	53%	0%	12%
$m=4$	13%	16%	25%	37%	19%	0%	21%	35%
$m=5$	24%	4%	14%	28%	72%	15%	13%	6%

New Solution 1

	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$	$i=6$	$i=7$	$i=8$
$m=1$	10%	3%	22%	0%	0%	32%	0%	47%
$m=2$	30%	15%	63%	20%	0%	0%	66%	0%
$m=3$	0%	21%	0%	15%	9%	53%	0%	12%
$m=4$	51%	0%	15%	37%	19%	0%	21%	35%
$m=5$	9%	61%	0%	28%	72%	15%	13%	6%

New Solution 2

	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$	$i=6$	$i=7$	$i=8$
$m=1$	63%	36%	51%	0%	44%	0%	0%	24%
$m=2$	0%	0%	0%	0%	31%	40%	88%	0%
$m=3$	0%	44%	0%	0%	0%	60%	0%	0%
$m=4$	13%	16%	25%	31%	0%	0%	0%	76%
$m=5$	24%	4%	14%	69%	25%	0%	12%	0%

(b) Two-point crossover.

Solution 1

	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$	$i=6$	$i=7$	$i=8$
$m=1$	10%	3%	22%	0%	44%	0%	0%	24%
$m=2$	30%	15%	63%	0%	31%	40%	88%	0%
$m=3$	0%	21%	0%	0%	0%	60%	0%	0%
$m=4$	51%	0%	15%	31%	0%	0%	0%	76%
$m=5$	9%	61%	0%	69%	25%	0%	12%	0%

Solution 2

	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$	$i=6$	$i=7$	$i=8$
$m=1$	63%	36%	51%	0%	0%	32%	0%	47%
$m=2$	0%	0%	0%	20%	0%	0%	66%	0%
$m=3$	0%	44%	0%	15%	9%	53%	0%	12%
$m=4$	13%	16%	25%	37%	19%	0%	21%	35%
$m=5$	24%	4%	14%	28%	72%	15%	13%	6%

New Solution 1

	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$	$i=6$	$i=7$	$i=8$
$m=1$	10%	3%	51%	0%	0%	0%	0%	24%
$m=2$	30%	15%	0%	20%	0%	40%	88%	0%
$m=3$	0%	21%	0%	15%	9%	60%	0%	0%
$m=4$	51%	0%	25%	37%	19%	0%	0%	76%
$m=5$	9%	61%	14%	28%	72%	0%	12%	0%

New Solution 2

	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$	$i=6$	$i=7$	$i=8$
$m=1$	63%	36%	22%	0%	44%	32%	0%	47%
$m=2$	0%	0%	63%	0%	31%	0%	66%	0%
$m=3$	0%	44%	0%	0%	0%	53%	0%	12%
$m=4$	13%	16%	15%	31%	0%	0%	21%	35%
$m=5$	24%	4%	0%	69%	25%	15%	13%	6%

Fig. 5. Typical example of crossover policy.

iteration. A solution whose evaluation function outperforms the dominant solution fitness is replaced with the dominant solution after implementing the exchange policy. This policy prevents the algorithm from trapping in local solutions.

Revolution policy: A revolution policy tries to generate a new solution as the mutation operator does. A limited number of individual solutions are chosen and the commonly utilized operators, i.e. *insertion*, *inversion*, and *swap* are implemented in the proposed multi-objective evolutionary algorithm. Since these operators are well known and widely used in the development of heuristic and meta-heuristic algorithms, we provide graphical illustrations in Figs. A1–A3 in the Appendix. We refer the interested readers to Goldberg (1989) for further information.

4.1.4. Non-dominated solution evaluation

In this algorithm, each non-dominated solution with relative dominated solutions is grouped into a cluster. All clusters are then evaluated in each iteration of the algorithm, and finally, a cluster with a more desirable objective function value is positioned at the upper front. The cluster, whose position is supposed to be substituted, moves down to the lower front. The cluster placed at the lowest front is then removed from the solution pool and is no longer considered in the algorithm. In the following, we describe different steps of the considered cluster evaluation procedure. In Step 1, the evaluation function of each cluster, which is mainly influenced by the quality of the non-dominated solutions are calculated. This function is formulated as $\mathfrak{E}_m(x_\mathfrak{n}, x_r) = f''_m(x_\mathfrak{n}) + \varphi\sum_{r\in\mathfrak{C}} f''_m(x_r)/|\mathfrak{C}|$, for each objective function m and dominant solutions $x_\mathfrak{n}$, $\mathfrak{n}\in\mathfrak{N}$, where φ is given as a coefficient of the evaluation function with a positive value. In Step 2, we determine dominant solutions $x_\mathfrak{n}$ for each objective function m by using the evaluation function of each cluster found in Step 1. It should be noted that a bigger value of this coefficient gives more importance to the non-dominated solutions. After determining the evaluation function of each cluster, we calculate the evaluation function of each non-dominated solution, which is denoted by $\mathfrak{q}_\mathfrak{n}$, and the corresponding normalized value, which is denoted by $\mathfrak{Q}_\mathfrak{n}$ in Steps 3 and 4, respectively. The solutions in the first Pareto front of each cluster are then recorded in the archive list, as indicated in Step 5. The algorithm can be stopped after a predefined iteration though. This procedure is given in Algorithm 2.

Algorithm 2. (*Cluster evaluation procedure*)

```
INPUT: $\mathfrak{P}_\mathfrak{n}, \mathfrak{n} = 1, \cdots, |\mathfrak{N}|; f''_m(x_\mathfrak{n}); f''_m(x_r)$
Step 1:    for $m = 1, \cdots, |\mathscr{M}|$ do
             for $\mathfrak{n} = 1, \cdots, |\mathfrak{N}|$ do
               $\mathfrak{E}_m(x_\mathfrak{n}, x_r) = f''_m(x_\mathfrak{n}) + \varphi\sum_{r\in\mathfrak{C}} f''_m(x_r)/n$ end for
             end for
Step 2:    for $m = 1, \cdots, |\mathscr{M}|$ do
             $\mathfrak{E}^*_m = \max_{\mathfrak{n}\in\mathfrak{N}}\{\mathfrak{E}_m(x_\mathfrak{n}, x_r)\}$ end for
```

(continued on next page)

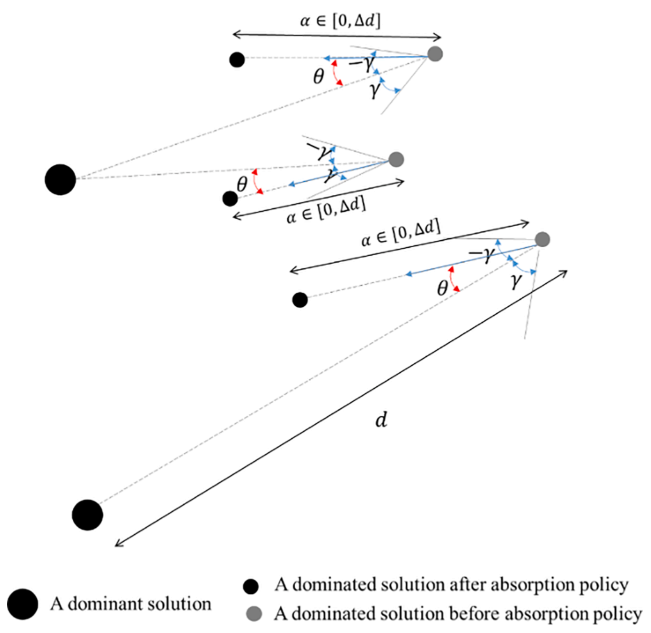
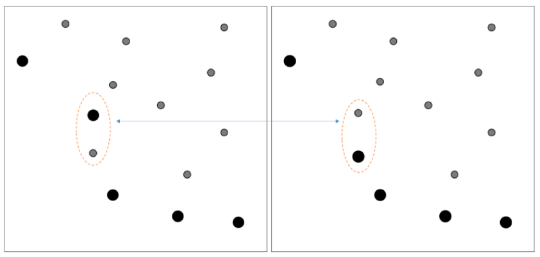

$\alpha \in [0, \Delta d]$
θ
$-\gamma$
γ
d
A dominant solution
A dominated solution after absorption policy
A dominated solution before absorption policy

Fig. 6. Typical example of an absorption policy.



(a) Before exchange. (b) After exchange.

Fig. 7. Exchange policy.

(*continued*)

```
Step 3:    for $\mathfrak{n} = 1, \cdots, |\mathfrak{N}|$ do
               $\mathfrak{q}_{\mathfrak{n}} = \sum_{m}^{|\mathscr{M}|} \mathfrak{E}_{m}^{*} - \mathfrak{E}_{m}(\mathbf{x}_{\mathfrak{n}}, \mathbf{x}_{i})$ end for
Step 4:    for $\mathfrak{n} = 1, \cdots, |\mathfrak{N}|$ do
               $\mathfrak{Q}_{\mathfrak{n}} = \mathfrak{q}_{\mathfrak{n}} / \sum_{\mathfrak{n}=1}^{|\mathfrak{N}|} \mathfrak{q}_{\mathfrak{n}}$ end for
Step 5:    SORT $\mathfrak{Q}_{\mathfrak{n}}$ and CHOOSE the best one as the first front cluster.
OUTPUT: Ranking of the best (most optimum) cluster.
```

Finally, we represent the proposed multi-objective evolutionary algorithm in the form of pseudocode illustration in Algorithm 3. The algorithm begins with fitting input parameters indicated by $\pi = \{\pi_1, \pi_2, \pi_3, \pi_4, \pi_5\}$ (i.e., number of the initial population, number of dominant solutions, percentages of crossover, percentages of absorption, and percentages of revolution, respectively). The set of input parameters (i.e., π) is represented in Appendix A. Accordingly, a set of initial solutions is generated regarding the procedure in Section 4.1.1 and Figs. 3–4. For each initial solution, the crowding distance (Algorithm 1) and fitness function are computed to rank them and form initial dominant solutions. When solutions are divided into dominant and dominated solutions, the clusters are formed and evolutionary policies, including crossover, absorption, exchanging, and revolution, as explained in Section 4.1.3, are executed to improve the quality of each cluster.

Algorithm 3. (*Pseudocode of the proposed algorithm*)

```
INPUT: terminate condition; input parameters $\pi$
GENERATE initial solutions
```

(*continued on next column*)

(*continued*)

```
EVALUATE fitness function for each individual solution
FORM initial dominant solutions:
While the terminate condition is not met do
    CALL the evolutionary policies
    CALL crowding distance procedure
    CALL cluster evaluation procedure
    UPDATE Archive
end while
OUTPUT: Non-dominated solutions.
```

Afterward, the quality of each cluster is evaluated according to Algorithm 2, based on which they are positioned in front 1, front 2, and so on. The cluster that is placed at the lowest front is then removed from the solution pool and its elements are assigned to other clusters. The evolutionary policies are iteratively implemented to search for a better result with neighborhood solutions. The procedure continues until the stopping criterion is met.

4.2. *Benchmark algorithms*

To obtain a benchmark for the performance of the proposed algorithm, a non-dominated sorting genetic algorithm (NSGA-II) and a Pareto-archived evolution strategy (PAES) algorithm are implemented for the instance test problems. NSGA-II is one of the most well-known and efficient multi-objective evolutionary algorithms. The objective of the NSGA-II is to generate a computationally fast and elitist population of candidate solutions and to improve the adaptive fit of them to a Pareto front constrained by a set of objective functions. It has been shown that NSGA-II can converge to the true Pareto-optimal front. A set of hierarchical sub-populations based on the ordering of Pareto dominance is created and similarity between members of each sub-population is measured on the Pareto front. The resulting groups and similarity measures are used to promote a diverse front of non-dominated solutions. Pareto archived evolution strategy (PAES) is another multi-objective evolutionary algorithm that works with a simple local search strategy to find diverse solutions in the Pareto optimal set. This algorithm controls the crowding of solutions in a deterministic search space by keeping diversity among the candidate solutions.

NSGA-II and PAES are usually considered as baseline algorithms for solving multi-objective optimization problems. While NSGA-II can converge faster and closer to the true Pareto-optimal front, PAES can improve the diversity of solutions in the obtained non-dominated front. We consider them as the benchmark algorithms and evaluate the performance of the proposed algorithm compared to the results obtained from NSGA-II and that of PAES. Since both NSGA-II and PAES have received well recognition in solving multi-objective optimization problems, we avoid explaining them and instead refer to Deb (2000, 2002) and Knowles and Corne (1999), respectively, for further information. It should be noted that algorithms are implemented in Matlab R2010 and performed on a PC with a Core i5 Duo CPU processor and 4 GB of RAM.

5. Numerical experiments

In this section, we present the results of two sets of instances that are designed for testing the effectiveness of the algorithms under investigation. The first set of instances is borrowed from the literature that addresses the network of a multiple-visit VRP (Dror, Laporte, & Trudeau, 1994). We compare the performances of the proposed solution method against the PAES and NSGA-II in terms of solution quality metrics. The second set of instances is taken from a hypothetical case study to capture the key managerial aspects of this problem.

5.1. *Test problems from the literature*

This section represents the design of test problems and the metrics used for comparing the performance of the algorithms introduced in Section 4.

5.1.1. Experimental plan

Test problems provided in the literature do not provide sufficient numerical values for all input parameters required for the problem formulated in this paper; therefore, we design the parameters of the test problems by expanding the available data used in the relevant literature. Following Xu et al. (2017), in the preliminary test, we have generated a wide range of problem instances, which vary in terms of the number of demand points, number of supply points, and number of product types in a transportation network. We have considered a transportation network with $N = \{10, 15, 20, 25, 30, 35, 40, 45, 50\}$ and $R = \{1, 2, 3, \cdots, 10\}$ and generated 90 different sized test problems. However, after testing all instances, each with two vehicle-duration time horizon, we realized that our proposed heuristic algorithm produces efficient results (in terms of multi-objective performance metrics, which will be explained in the next subsection) for the following test problems (N,R) = (10,3), (10,5), (10,8), (30,3), (30,5), (30,8), (50,3), (50,5), (50,8). It should be noted that these test problems reflect the real problems with a relatively large size. Therefore, we focus on a network structure with $N = \{10, 30, 50\}$ nodes, including demand points and supply points, which are randomly dispersed over a typical region. In this data set, the number of demand points is given in $D = \{6, 20, 35\}$ and the number of supply points is given in $P = \{4, 10, 15\}$. For each test problem, the number of product types distributed through the cross-dock network is indicated as the set $R = \{3,5,8\}$. Demand and supply quantities are generated following the way used in Dror et al. (1994) to create a condition in which nodes can be met by more than one vehicle. For each instance, the total loaded (for suppliers) and unloaded (for retailers) quantities are generated based on a uniform distribution in [αQ, βQ]; where α and β are control parameters and they are set to 0.1 and 0.5, respectively. Then, we verify the generated data by the method used in Chen et al. (2014) and Xu et al. (2017) to ensures that the supply is no fewer than the demand for each product within the transportation network.

As far as the heterogeneous vehicle fleets are concerned in this paper, a specific level of carrying capacity is assigned to inbound and outbound vehicles within the range of $[150, 250]$ units. The total number of vehicles used for each instance, i.e. V, belongs to the interval [$\overline{V}$, N], where $1 \leq \overline{V} \leq \mathrm{N}$. The value of $\overline{V}$ represents the minimum number of vehicles calculated by $\lceil\frac{\sum_{i\in D}\sum_{r}\mathrm{f}_r \times d_{ir}}{d}\rceil$, where $\lceil\cdot\rceil$ denotes the rounded-up value, and N is the maximum value of the number of pickup and delivery nodes, i.e. $\max\{P, D\}$. According to Xu et al. (2017), the average speed of each truck is supposed to be 60 km/h. In the network we designed, the inbound and outbound transportation time of each vehicle depending on the distance is distributed within $[20, 200]$ minutes. The corresponding transportation cost per kilometer is estimated at \$1.8 and \$0.8 for inbound and outbound vehicles, respectively. Loading or unloading time at each node is randomly generated by a uniform distribution function $U\ [10, 80]$ in a minute per unit pallet.

Time windows in the VRPCD are slightly different from the VRPs since the synchronization of vehicle deployment is also dependent on the operations at the cross-dock; that is, the delivery process cannot be started until picking-up and cross-docking operations end. The time windows are generated by adopting the approach used in Bin (2006) in which it is assumed that a vehicle reaches the first pickup node in one hour after leaving the cross-dock and that a vehicle requires one hour going back to the depot. We denote the time to visit a node by the following format$HH : MM$, where HH and MM represent the hours and minutes, respectively. We also consider two time-intervals for vehicle-duration in pickup and delivery operations for each instance. In the first vehicle-duration set, pickup and delivery processes are assumed to last 6 and 10 h, respectively. In the second one, pickup and delivery duration time are set to 8 and 8 h, respectively. For the ease of presentation, the first and the second vehicle-duration sets are denoted by *6H–10H* and *8H–8H*, respectively. In the *6H–10H* time structure, the latest time to visit a supplier and a retailer is randomly generated following the uniform distribution function $U\ [03:00, 07:00]$ and $U\ [13:00, 17:00]$, respectively. In the *8H–8H* time structure, the latest time to visit a supplier and a retailer is randomly generated using the uniform distribution function $U\ [04:00, 09:00]$ and $U\ [15:00, 17:00]$, respectively. In both time structures, we assume that the earliest time to visit a node is one hour in advance of the corresponding latest time. The total duration time of pickup and delivery processes for each time structure is set to $T = 18$ h.

Different values of N and R in the time structures of *6H–10H* and *8H–8H* constitute each scale for test problems. To overcome randomness and improve computational experiments, we randomly generate 5 instances for each scale and testing the results. This process leads to totally 90 instances.

We calibrate the parametric inputs, i.e., number of the initial population, number of dominant solutions, percentages of crossover, percentages of absorption, and percentages of revolution, of the proposed algorithm by Design Expert software, using the response surface methodology (RSM), as described in Appendix A.

5.1.2. Multi-objective performance metrics

In this section, we represent a systematic comparison of the performance of the proposed evolutionary algorithms using the following four metrics: *Quality metric, Distance to the covered space*, *Extent of the covered space*, and *Spacing metric*. These metrics are commonly used in multi-objective optimization experiments for measuring convergence and diversity of Pareto front solutions (Tavakkoli-Moghaddam, Azarkish, & Sadeghnejad, 2011).

1) *Quality metric (QM)*: This metric counts the number of non-dominated solutions that can be obtained by each algorithm. To measure it, the number of non-dominated solutions belonging to each algorithm is compared to the total number of non-dominated solutions found from the counter algorithms studied in this paper, i.e. PAES and NSGA-II. In other words, if all of the non-dominated solutions belong to a given algorithm, *QM* returns 1 (or 100%). A higher quality metric means better performance.
2) *Distance to the covered space* ($\mathscr{D}_{\mathrm{cs}}$). This metric, which relates to the Hyper-volume metric or $\mathscr{S}$-metric proposed by Zitzler and Thiele (1999), is extensively used to measure convergence and diversity of the produced Pareto solutions. We use such a metric to calculate the objective space covered by the non-dominated set of solutions. However, this metric requires a known or an approximate Pareto optimum front. Because no true Pareto is known for this problem, we consider the best Pareto solutions found by all algorithms investigated in this paper as the true Pareto front. Let $\mathscr{X} = (x_1, x_2, \cdots, x_k)$ be a set of Pareto solutions containing k variables. Let us recall that $\mathrm{f}_{m}^{p}(\mathrm{x}_i)$ represents the m-th objective function value of the set of solution $x_i, i = 1, \cdots k$ of the Pareto front p. The best and the worst Pareto solutions are estimated by $\min_{p}\{\mathrm{f}_{m}^{p}(\mathrm{x}_i)\}$ and $\max_{p}\{\mathrm{f}_{m}^{p}(\mathrm{x}_i)\}$, respectively. We denote the distance to the covered space by $d_m(x_i)$, represented in Equation (25), which gives the normalized value of the distance of a non-dominated solution set x_i associated with the m-th objective to the corresponding closest point in the estimated true Pareto front.

$$d_m(x_i) = \frac{f_m(x_i) - \min_{p}\{f_m^{p}(x_i)\}}{\max_{p}\{f_m^{p}(x_i)\} - \min_{p}\{f_m^{p}(x_i)\}} \quad \forall i = 1, \cdots k, m \in \mathscr{M} \tag{25}$$

Then, we take the idea of root-square of the normalized distances, introduced in Van Veldhuizen and Lamont (1998), to measure the Euclidian distance between each of the non-dominated solutions and the best Pareto front, here denoted by $f_m^{*}(x_i) = \min_{p}\{f_m^{p}(x_i)\}$, found from all algorithms studied in this paper. Mohammadi, Jolai, and Tavakkoli-Moghaddam (2013) and Lwin, Qu, and Kendall (2014) also showed that this metric efficiently measures both the diversity and the convergence of the non-dominated solutions found by the algorithm. This metric is formulated by:

$$\mathscr{D}_{cs} = \frac{1}{k}\sum_{i=1}^{k}\sqrt{\sum_{m\in\mathscr{M}}(d_m(\mathrm{x}_i))^2} \tag{26}$$

Concerning the formulation of $\mathscr{D}_{cs}$, the smaller value of the distance of the covered space is preferred as it shows that the non-dominated solutions are closer to the true Pareto front set.

3) *Extent of the covered space* ($\mathscr{E}x_{cs}$). This metric is important to achieve simultaneously a well-distributed and well-spread non-dominated front (Deb, 1999). It measures the range of values covered by the non-dominated solutions for m-th objective, as represented by:

$$ex_m = \frac{\max_{i,p}\{f_m^p(x_i)\} - \min_{i,p}\{f_m^p(x_i)\}}{\max_{p}\{f_m^p(x_i)\} - \min_{p}\{f_m^p(x_i)\}}\forall m \in \mathscr{M} \tag{27}$$

where ex_m shows the extent of the obtained non-dominated fronts for each objective (Zitzler, Deb, & Thiele, 2000). This metric is then incorporated in the inverted generational distance (IGD), which is proposed by Czyzzak and Jaszkiewicz (1998) and later developed by Sierra and Coello Coello (2005), to calculate the level of extent of the solutions covered by all objectives obtained from an algorithm. The root-mean-square of the level of extent of objectives is represented by:

$$\mathscr{E}x_{cs} = \sqrt{\sum_{m\in M} ex_m^2} \tag{28}$$

It is important to mention that the higher value of the extent metric ($\mathscr{E}x_{cs}$) indicates a wider range of space covered by the non-dominated solutions. It is to be maximized.

4) *Spacing metric* ($\mathscr{S}p$). This metric quantifies the uniformity of the spread of a non-dominated set of solutions using the average function of the difference between the spacing of consecutive points and the mean spacing of consecutive points (Deb, Agrawal, Pratap, & Meyarivan, 2000). The spread of a non-dominated set of solutions is measured by the Euclidean distance from the set of Pareto solutions $\mathscr{X} = (x_1, x_2, \cdots, x_k)$ to the closest Pareto-optimal solutions obtained from the total enumeration of it. As discussed before, we apply the estimated minimum values of the mth objective function, indicated by $\mathrm{f}_m^*(x_i) = \min_{p}\{f_m^p(x_i)\}$, to represent the true Pareto solutions obtained from all algorithms investigated in this paper. Therefore, the Euclidean distance between the set of Pareto solutions $\mathscr{X} = (x_1, x_2, \cdots, x_k)$ and the best-estimated Pareto-optimal solutions are as follows:

$$\mathrm{Dis}^*(x_i) = \sqrt{\sum_{m\in M}(f_m^p(x_i) - f_m^*(x_i))^2}. \tag{29}$$

This metric, which has widely been used in the evaluation of the performance metrics in multi-objective optimization problems (Mohtashami, Tavanab, Santos-Arteagad, & Fallahian-Najafabadi, 2015; Tavakkoli-Moghaddam et al., 2011), considers the average discrepancy of the distance of consecutive obtained solutions and the average solutions as represented by:

$$\mathscr{S}p = \frac{\sum_{i=1}^{k}\left|Dis^*(x_i) - \overline{Dis}^*\right|}{(k-1)\overline{Dis}^*}, \tag{30}$$

where

$$\overline{Dis}^* = \frac{1}{k}\sum_{i=1}^{k} Dis^*(x_i) \tag{31}$$

The smaller the value of this metric is, the more uniformity and convergence the obtained front has.

5.1.3. *Results analysis*

We execute the proposed algorithms for each instance of the test problem with different numbers of suppliers and retailers and random data 10 times and report the performance metrics over the number of replications in Tables 1 and 2. As indicated in these tables, the proposed multi-objective evolutionary algorithm (MOEA) strongly outperforms the PAES and NSGA-II in terms of *quality metric*, with the average value of 0.93 against 0.00 and 0.07 in pickup–delivery duration *6H–10H* and 0.91 against 0.00 and 0.09 in pickup–delivery duration *8H–8H*. Summary statistics (e.g., standard deviation, minimum and maximum of algorithms performance) are reported in Table 3 in terms of four above-mentioned metrics. As shown in this table, for the quality metric, the NSGA-II cannot reach more than 0.50 value over the instances and PAES is unable to find more efficient non-dominated solutions than the other algorithms we studied. We also observe that the distance to the covered space metric, which represents the proximity to the true Pareto front set, corresponding to the proposed MOEA is, on average, 0.290.32 compared to 0.6 and 1.07, corresponding to the NSGA-II and PAES, respectively in the *6H–10H* time structure, while is 0.33 against 0.56 and 1.11 in the *8H–8H* time structure. The proposed MOEA experienced a more interesting extent of the covered space metric with a value of 0.79 against the NSGA-II and PAES with 0.70 and 0.46, respectively in the first time structure and 0.76 compared to 0.69 and 0.46 corresponding to the NSGA-II and PAES in the second time structure. It can be seen that the performance of the proposed MOEA from the spacing metric viewpoint is more efficient than NSGA-II but not as good as PAES.

In general, results show that the proposed MOEA outperforms the NSGA-II and PAES in most metrics discussed here and that NSGA-II can return better non-dominated solutions in comparison to PAES for some of the metrics considered in this paper. Experimental tests show that as the size of the test problems increases, the operational cost and the violation from the promised time tends to increase. In such cases, we observe a reduction in the quality of Pareto solutions found by the benchmark algorithms from test size N = 30 to N = 50, while the algorithm we developed tends to maintain its quality over the different sized instances. Regarding Table 3, in all problem scales, the quality metric of MOEA is higher when pickup–delivery duration is *6H–10H* comparing to 8*H–8H* pickup–delivery duration. Finally, we give an illustrative presentation of the Pareto solutions found by the algorithms studied in this paper for some typical instances in Fig. 8(a)–(f).

A small-sized instance is presented here for evaluating the solution of two objective values and the routes obtained using the mathematical model to support our proposed algorithm. This instance consists of three suppliers (P1, P2, and P3), three retailers (D1, D2, and D3), a cross-docking center (CD), two types of products (R1 and R2), and four available vehicles. We consider two types of time windows in this instance: narrow and wide. We apply the following multi-objective optimization methods for further analysis.

The first one is the augmented epsilon-constraint (AEC) method that takes one objective function to optimize and convert the rest as constraints. It reaches to the Pareto front in an iterative procedure. This method is a useful one to find the Pareto front in multi-objective problems (Vafaeenezhad, Tavakkoli-Moghaddam, & Cheikhorouhou, 2019). The second one is a simple weighted summation (SWS) method. This method creates an integrated objective function giving a weight to each objective. The main objective function is the weighted summation of the original ones. The third one is the normalized weighted summation (NWS) method, which works like SWS.

The difference is in creating the integrated objective function. As the

Table 1
Performance metrics of the algorithms for pickup–delivery duration *6H–10H*.

Instance size				Quality metric (*QM*)			Distance to covered space ($\mathscr{D}_{cs}$)			Extent of the covered space ($\mathscr{E}x_{cs}$)			Spacing metric ($\mathscr{S}p$)		
N	*D*	*P*	*R*	NSGA-II	PAES	MOEA	NSGA-II	PAES	MOEA	NSGA-II	PAES	MOEA	NSGA-II	PAES	MOEA
10	6	4	3	0	0	1*	0.79	1.23	0.23*	0.53	0.3	0.67*	0.82	0.44*	0.91
				0	0	1*	0.69	1.26	0.10*	0.57*	0.31	0.23	0.74	0.71*	0.74
				0.17	0	0.83*	0.22	1.23	0.15*	0.46*	0.32	0.32	0.61*	0.59	0.63
				0	0	1*	0.80	1.33	0.23*	0.65*	0.23	0.58	0.72	0.86	0.63*
				0	0	1*	0.59	1.00	0.14*	0.81*	0.15	0.27	0.91	0.60	0.47*
10	6	4	5	0.15	0	0.85*	0.40	1.08	0.36*	0.73	0.86*	0.73	1.79	0.51	0.5*
				0	0	1*	0.48	1.18	0.08*	0.30	0.48*	0.23	0.39*	0.66	0.88
				0.31	0	0.69*	0.28	0.95	0.26*	1.04*	0.46	0.45	0.86	0.65	0.63*
				0	0	1*	0.43	1.24	0.18*	0.60*	0.33	0.45	0.68	0.58*	0.68
				0.1	0	0.9*	0.38	1.23	0.26*	0.91*	0.3	0.35	0.75	0.53*	0.55
10	6	4	8	0	0	1*	0.43	1.04	0.37*	0.65	0.76	0.85*	1.65	0.93	0.27*
				0	0	1*	0.36	1.23	0.13*	0.33	0.31	0.38*	0.76	0.64*	0.66
				0.07	0	0.93*	0.33	1.26	0.18*	0.54*	0.25	0.45	0.57	0.16*	0.97
				0	0	1*	0.47	1.16	0.15*	0.41	0.51*	0.26	0.60*	0.70	0.71
				0	0	1*	0.46	1.15	0.13*	0.42	0.46*	0.35	0.79	0.43*	0.65
30	20	10	3	0	0	1*	0.49	1.02	0.24*	1.04*	0.33	06	0.98	0.82	0.64*
				0	0	1*	0.34	1.14	0.30*	0.06	0.58	0.85*	0.10*	0.83	0.97
				0.33	0	0.67*	0.48	1.02	0.40*	1.07*	0.82	0.79	0.71	0.68*	0.90
				0.41	0	0.59*	0.58	1.05	0.38*	1.31*	0.42	0.66	0.91	0.36*	0.42
				0	0	1*	0.52	0.94	0.15*	1.10*	0.76	0.31	0.70	0.45*	0.89
30	20	10	5	0	0	1*	0.88	0.45	0.22*	0.92*	0.33	0.51	0.99	0.5	0.41*
				0	0	1*	0.66	1.07	0.24*	0.44	0.58	0.59*	0.69*	0.79	0.87
				0	0	1*	0.70	1.22	0.32*	0.54	0.34	0.95*	0.83*	1.17	0.88
				0.25	0	0.75*	0.29	1.20	0.24*	0.41	0.43	0.68*	0.11*	0.96	0.97
				0	0	1*	0.44	1.19	0.24*	0.50	0.46	0.79*	1.05	0.71*	0.78
30	20	10	8	0	0	1*	0.94	0.75	0.19*	0.72*	0.30	0.47	1.24	0.55	0.3*
				0	0	1*	0.79	1.19	0.42*	1.05	0.40	1.18*	0.46*	0.70	0.65
				0	0	1*	0.36*	0.97	0.45	0.51	0.76	1.27*	0.40*	0.78	0.75
				0.17	0	0.83*	0.67	0.76	0.45*	1.17*	0.64	0.87	0.74	0.45*	0.78
				0	0	1*	0.49	1.20	0.22*	1.09*	0.44	0.55	0.66	0.13*	0.43
50	35	15	3	0	0	1*	0.93	0.91	0.56*	0.63	0.68	1.15*	0.82	0.59	0.5*
				0	0	1*	0.50	1.15	0.36*	0.75	0.28	0.82*	0.62	0.52	0.37*
				0	0	1*	0.75	1.19	0.37*	0.72	0.34	1.10*	0.16*	0.62	0.94
				0	0	1*	0.69	1.18	0.18*	0.56	0.36	0.69*	0.61	0.41	0. 37*
				0	0	1*	0.84	1.21	0.33*	0.54	0.46	0.72*	0.66*	0.83	0.77
50	35	15	5	0	0	1*	0.92	0.29*	0.29*	0.61	0.43	0.86*	0.73*	1.14	1.29
				0	0	1*	0.64	1.06	0.35*	0.73*	0.64	0.63	0.71	0.37*	0.73
				0.41	0	0.58*	0.61*	1.07	0.64	1.13*	0.46	1.06	0.86	0.41*	0.63
				0	0	1*	0.64	1.15	0.32*	0.77*	0.53	0.62	0.80	0.47*	0.64
				0.17	0	0.83*	0.55	1.00	0.54*	0.98	0.76	1.15*	0.97	0.68*	0.68*
50	35	15	8	0	0	1*	1.1	0.98	0.43*	0.63	0.47	0.92*	0.62	0.72	0.23*
				0	0	1*	0.91	1.19	0.46*	0.51	0.32	0.85*	0.64	0.33*	0.69
				0.5*	0	0.5*	0.50	1.06	0.39*	0.60	0.57	0.64*	0.82	0.69	0.51*
				0	0	1*	0.84	1.30	0.37*	0.47	0.34	0.83*	0.78	0.68*	0.75
				0	0	1*	0.72	0.90	0.27*	1.11*	0.46	0.51	0.56	0.92	0.44*
Average				0.07	0.00	**0.93***	0.60	1.07	**0.29***	0.70	0.46	**0.79***	0.75	**0.63***	0.67

The best output of each metric in each problem instance is marked with *.

original objective functions may have different scales, they are first normalized (by finding the best and worst value of them) and then a weighted summation of them is applied. The employed AEC is run for 26 iterations. Comparing the algorithms, SWS and NWS are also run for 26 sets of weights. In all methods, the final 26 solutions are compared to each other and the non-dominated solutions are selected as the points of the Pareto front. Tables 4 and 5 report the points of obtained Pareto fronts for narrow and wide time windows, respectively. Each table shows the optimum routes, the methods by which the solution is obtained, and the obtained objective values, where OFV_1 gives minimum total costs while OFV_2 represents the possible violations from the given time window. We can observe that the AEC can find all Pareto points while NWS and SWS can only find some of them.

5.2. Case problem: Walmart cross-docking in Houston, Texas

The second set of numerical tests is dedicated to solving the problem studied in this paper with public data from Walmart distribution network. The geographical scope of our study is limited to Walmart outlets in Houston, Texas. We choose 20 largest Walmart stores in the proximity suitable for cross-docking in Houston as demand points with geographical location available at www.walmart.com. The corresponding demand has been retrieved from the sales data of Walmart outlets in Houston (www.kaggle.com). We consider the perishables distribution center, as the cross-dock facility in Walmart distribution network in Houston, Texas, to consolidate the perishable category of products (which are dairy products, fruits and vegetables, and bread and bakery products in this case study). According to www.mwpvl.com, we choose three Walmart's full-line distribution centers in Houston, Texas, as main suppliers, to provide the category of products. We assume that 9-ton medium/heavy-duty trucks and 4-ton light-duty trucks are utilized in pickup operations and delivery operations, respectively, in the transportation fleet. Cross-docking operations are assumed to take a 5-h work time during 22 h of time horizon, as well. The other input data related to pickup and delivery operations are represented in Table 6, which are partially retrieved from www.atri-online.org.

For this problem setting, we consider two scenarios. The first one is our baseline scenario, which refers to a single-visit VRPCD. The second one is the model optimized in this paper and recognizes order splitting. To solve this problem, we apply the proposed MOEA, whose

Table 2
Performance metrics of the algorithms for pickup–delivery duration *8H–8H*.

Instance size				Quality metric (*QM*)			Distance to covered space ($\mathscr{D}_{cs}$)			Extent of the covered space ($\mathscr{E}_{x_{cs}}$)			Spacing metric ($\mathscr{S}_p$)		
N	*D*	*P*	*R*	NSGA-II	PAES	MOEA	NSGA-II	PAES	MOEA	NSGA-II	PAES	MOEA	NSGA-II	PAES	MOEA
10	6	4	3	0	0	1*	0.79	1.23	0.23*	0.53	0.3	0.67*	0.82	0.91	0.44*
				0.45	0	0.55*	0.23*	1.10	0.28	0.31	0.57*	0.50	0.64*	0.77	1.38
				0	0	1*	0.31	1.14	0.13*	0.28	0.46*	0.27	0.71	0.15*	1.68
				0	0	1*	0.62	1.24	0.12*	0.65*	0.31	0.22	1.80	0.36*	0.64
				0	0	1*	0.21	1.25	0.13*	0.39*	0.31	0.25	1.62	0.58	0.34*
10	6	4	5	0	0	1*	0.73	1.07	0.21*	0.35	0.59*	0.41	1.79	0.4*	0.64
				0	0	1*	0.43	1.00	0.20*	0.92	0.21	1.03*	1.05	0.65*	1.22
				0.07	0	0.93*	0.57	0.90	0.25*	0.99*	0.19	0.57	1.19	0.41*	0.57
				0.12	0	0.88*	0.45	1.13	0.29*	0.77*	0.51	0.50	0.65	0.47*	0.75
				0	0	1*	0.57	1.11	0.33*	1.01*	0.02	0.77	0.63	0.26*	0.59
10	6	4	8	0.43	0	0.57*	0.51	1.15	0.32*	0.51	0.54	0.65*	1.11	0.39*	0.39*
				0	0	1*	0.69	1.10	0.18*	0.66	0.86*	0.43	0.56*	1.07	0.72
				0.17	0	0.83*	0.29	1.03	0.26*	0.47	0.28	1.08*	0.72	0.71*	0.98
				0	0	1*	0.74	1.07	0.14*	0.74*	0.39	0.30	0.54*	0.74	0.63
				0	0	1*	0.54	1.27	0.43*	0.79	0.24	0.99*	0.61	0.27*	0.58
30	20	10	3	0	0	1*	0.92	1.05	0.19*	0.67	0.42	0.76*	0.75*	1.05	0.75*
				0	0	1*	0.40	0.97	0.28*	1.00*	0.22	0.62	0.99	0.36*	0.36*
				0	0	1*	0.35*	1.18	0.40	0.44	0.41	1.01*	0.68*	0.90	0.71
				0	0	1*	0.22*	1.23	0.23	0.36	0.43	0.78*	0.89	0.72*	1.03
				0.41	0	0.59*	0.41	0.99	0.30*	1.00*	0.76	0.58	0.78	0.53*	0.74
30	20	10	5	0.27	0	0.73*	0.53	0.85	0.42*	1.01*	0.52	0.91	0.86	0.28*	0.54
				0	0	1*	0.37	1.19	0.16*	0.83*	0.47	0.30	0.81	0.36*	0.81
				0.25	0	0.75*	0.29*	1.25	0.46	0.48	0.34	0.95*	0.69	0.48*	1.01
				0	0	1*	0.80	1.24	0.36*	0.36	0.49	1.03*	0.51	0.27*	0.82
				0	0	1*	0.29*	1.16	0.64	0.65	0.62	1.07*	1.02	0.44*	0.58
30	20	10	8	0	0	1*	0.48	0.62	0.32*	0.63	0.75	1.08*	1.01	0.57*	1.3
				0.17	0	0.83*	0.63	1.08	0.36*	1.16*	0.54	0.83	0.80	0.35*	0.91
				0.33	0	0.67*	0.51*	0.71	0.59	0.75	0.88	1.30*	0.48	0.19*	0.77
				0	0	1*	0.66	1.25	0.34*	0.41	0.41	0.74*	0.69	0.25*	0.69
				0	0	1*	0.37*	1.15	0.45	0.61	0.53	1.11*	0.72	0.60*	0.89
50	35	15	3	0	0	1*	0.93	0.91	0.56*	0.63	0.68	1.15*	0.52	0.59	0.5*
				0	0	1*	0.70	1.20	0.37*	0.41	0.41	0.94*	0.57*	0.88	0.93
				0	0	1*	0.82	0.96	0.14*	0.59*	0.57	0.45	0.81	0.89	0.47*
				0	0	1*	0.72	1.06	0.32*	0.89	0.39	0.99*	0.79	0.86	0.43*
				0.17	0	0.83*	0.48	1.17	0.41*	0.48	0.53	0.76*	0.75 78	0.55*	0.68
50	35	15	5	0	0	1*	0.56	1.23	0.43*	0.78	0.4	0.79*	0.75	0.49*	0.82
				0	0	1*	0.68	1.08	0.50*	1.30*	0.49	1.12	0.42*	0.48	0.74
				0	0	1*	0.73	1.02	0.27*	0.66*	0.62	0.57	0.71	0.54*	0.71
				0.3	0	0.7*	0.34*	1.17	0.36	0.78	0.50	0.90*	0.78	0.71*	0.75
				0.29	0	0.71*	0.52	1.18	0.40*	0.95*	0.45	0.92	0.68*	0.77	0.82
50	35	15	8	0	0	1*	0.75	1.26	0.17*	0.4	0.4	0.47*	0.55*	0.61	0.94
				0	0	1*	0.89	1.26	0.37*	0.72	0.28	0.84*	0.36*	0.69	0.71
				0.44	0	0.56*	0.64	1.22	0.53*	0.90	0.42	1.03*	0.76	0.46*	0.71
				0	0	1*	0.86	1.17	0.45*	0.93*	0.36	0.87	0.66	1.16	0.36*
				0	0	1*	0.69	1.23	0.38*	0.68*	0.59	0.50	1.24	0.43*	0.69
Average				0.09	0	**0.91***	0.56	1.11	**0.33***	0.69	0.46	**0.76***	0.81	**0.56***	0.75

The best output of each metric in each problem instance is marked with *.

performance efficiency has been proved earlier. We then compare solutions to both the above-mentioned scenarios and finally report some managerial insights accordingly. Note that results are dependent on the input data used for this case study and may vary upon different cost categories.

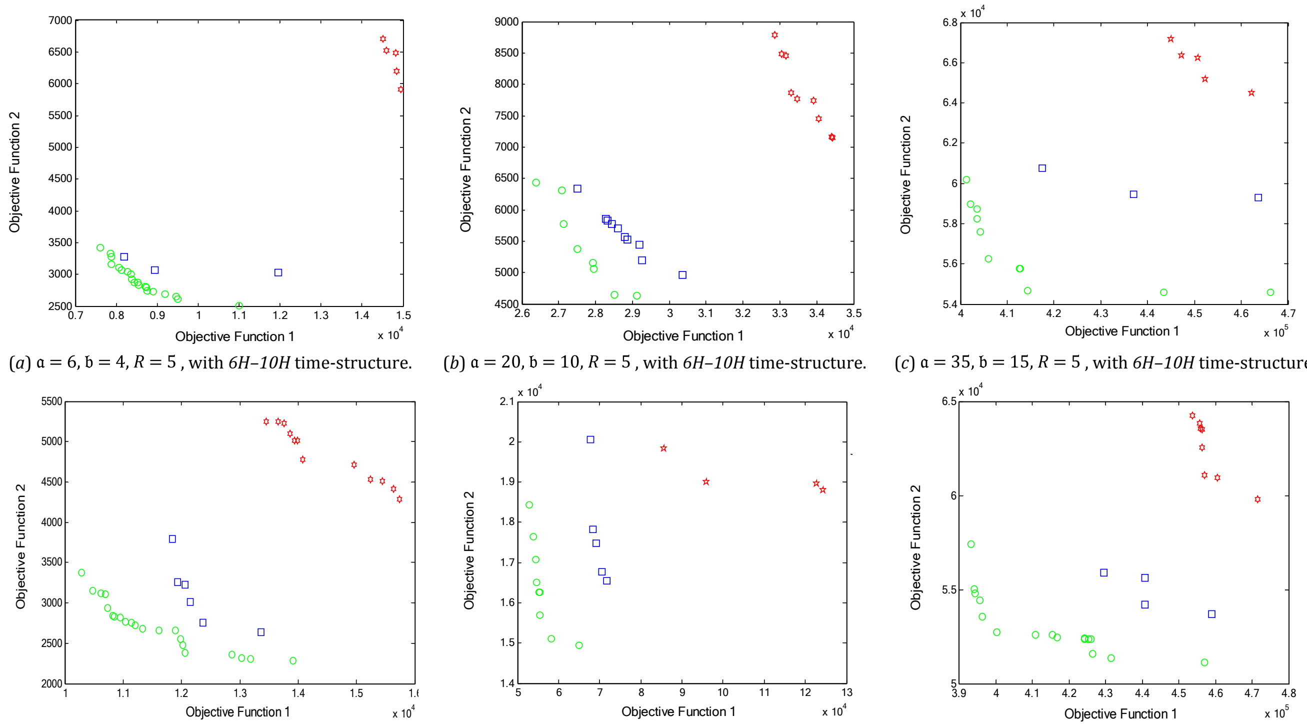
For each scenario, we consider two types of criteria for the case study data, which are efficiency and responsiveness. To analyze the results and highlight the improvements, we use the most common practical indicators in the retail sector and compare solutions of two considered scenarios for each criterion in Table 7. In the following, we present our discussion on the performance of the two considered scenarios.

For the efficiency criterion, we use the following indicators, total transportation and operation cost, truck capacity utilization, and total traveled distance. We observe that the baseline scenario, which does not recognize order-splitting, returns $9838.78 transportation cost, while the optimized one recognizing order-splitting ends with $9684.82 transportation cost, showing 1.56% improvement. However, it should be noted that the operational cost remains unchanged, as the optimum number of trucks to dispatch in both scenarios is the same. Overall, results reveal that the consideration of order splitting reduces the total transportation and operational cost by 1.54%. We also observe that when order splitting is recognized in the model, four medium/heavy-duty truck with a capacity utilization rate of 87.50%, on average, is used for pickup fleets in both scenarios. On the other hand, 10 light-duty trucks with a capacity utilization rate of 65.46% and 66.39%, on average, are used on delivery routes in the baseline and proposed scenarios, respectively. It shows that order splitting contributes to improving delivery fleet capacity utilization rate up to 1.40%, however it remains unchanged for the pickup routes. We also notice when order splitting is allowed, the total distance traveled by trucks in the distribution network is reduced by 1.72%. Results show that the total distance traveled in the pickup routes under the baseline scenario is 1801.76 km, while this metric under the optimized one returns 1722.99 km. This result shows a 4.37% reduction in pickup traveled distance when recognizing order splitting. We also observe that the total distance traveled in the delivery routes under the baseline scenario and the optimized one is 7009.50 km and 6937.13 km, indicating 1.03% reduction in the traveled distance in the delivery routes. The proposed model reduces the pickup route distance by 4.37% and the delivery distance by 1.03%, which confirms a reduction in transportation cost.

Table 3
Statistics of algorithms performance.

N	P–D duration	Statistics	Quality metric (QM)			Distance to the covered space ($\mathscr{D}_{cs}$)			Extent of the covered space ($\mathscr{E}_{x_{cs}}$)			Spacing metric ($\mathscr{S}_{p}$)		
			NSGA-II	PAES	MOEA	NSGA-II	PAES	MOEA	NSGA-II	PAES	MOEA	NSGA-II	PAES	MOEA
10	*6H–10H*	Average	0.05	0.00	0.95*	0.47	1.17	0.20*	0.60*	0.40	0.44	0.84	0.60*	0.66
		SD	0.09	0.00	0.09	0.17	0.10	0.08	0.20	0.19	0.18	0.37	0.18	0.17
		Min	0.00	0.00	0.69	0.22	0.95	0.08	0.30	0.15	0.23	0.39	0.16	0.27
		Max	0.31	0.00	1.00	0.80	1.33	0.37	1.04	0.86	0.85	1.79	0.93	0.97
	8H–8H	Average	0.08	0.00	0.92*	0.51	1.12	0.23*	0.62*	0.39	0.58	0.96	0.54*	0.77
		SD	0.15	0.00	0.15	0.19	0.10	0.09	0.24	0.21	0.29	0.45	0.26	0.38
		Min	0.00	0.00	0.55	0.21	0.90	0.12	0.28	0.02	0.22	0.54	0.15	0.34
		Max	0.45	0.00	1.00	0.79	1.27	0.43	1.01	0.86	1.08	1.80	1.07	1.68
30	*6H–10H*	Average	0.08	0.00	0.92*	0.58	1.00	0.30*	0.80	0.51	1.10*	0.70	0.66*	0.71
		SD	0.14	0.00	0.14	0.19	0.22	0.10	0.36	0.17	1.38	0.33	0.26	0.22
		Min	0.00	0.00	0.59	0.29	0.45	0.15	0.06	0.30	0.31	0.10	0.13	0.30
		Max	0.41	0.00	1.00	0.94	1.22	0.45	1.31	0.82	6.00	1.24	1.17	0.97
	8H–8H	Average	0.10	0.00	0.90*	0.48	1.06	0.37*	0.69	0.52	0.87*	0.78	0.49*	0.79
		SD	0.15	0.00	0.15	0.20	0.20	0.13	0.26	0.17	0.25	0.16	0.25	0.22
		Min	0.00	0.00	0.59	0.22	0.62	0.16	0.36	0.22	0.30	0.48	0.19	0.36
		Max	0.41	0.00	1.00	0.92	1.25	0.64	1.16	0.88	1.30	1.02	1.05	1.30
50	*6H–10H*	Average	0.07	0.00	0.93*	0.74	1.04	0.39*	0.72	0.47	0.84*	0.69	0.63*	0.66
		SD	0.16	0.00	0.16	0.18	0.24	0.12	0.21	0.14	0.21	0.19	0.22	0.26
		Min	0.00	0.00	0.50	0.50	0.29	0.18	0.47	0.28	0.51	0.16	0.33	0.23
		Max	0.50	0.00	1.00	1.10	1.30	0.64	1.13	0.76	1.15	0.97	1.14	1.29
	8H–8H	Average	0.08	0.00	0.92*	0.69	1.14	0.38*	0.74	0.47	0.82*	0.69	0.64*	0.68
		SD	0.15	0.00	0.15	0.16	0.11	0.12	0.24	0.11	0.23	0.21	0.27	0.17
		Min	0.00	0.00	0.56	0.34	0.91	0.14	0.40	0.28	0.45	0.36	0.00	0.36
		Max	0.44	0.00	1.00	0.93	1.26	0.56	1.30	0.68	1.15	1.24	1.16	0.94

The best output of each metric in terms of average value of each problem set is marked with *.



(a) $\mathfrak{a} = 6$, $\mathfrak{b} = 4$, $R = 5$, with *6H–10H* time-structure. (b) $\mathfrak{a} = 20$, $\mathfrak{b} = 10$, $R = 5$, with *6H–10H* time-structure. (c) $\mathfrak{a} = 35$, $\mathfrak{b} = 15$, $R = 5$, with *6H–10H* time-structure.

(d) $\mathfrak{a} = 6$, $\mathfrak{b} = 4$, $R = 5$, with *8H–8H* time-structure. (e) $\mathfrak{a} = 20$, $\mathfrak{b} = 10$, $R = 5$, with *8H–8H* time-structure. (f) $\mathfrak{a} = 35$, $\mathfrak{b} = 15$, $R = 5$, with *8H–8H* time-structure.

Fig. 8. Non-dominated solutions found by the algorithms (☆ PAES; □ NSGA-II; ○ MOEA).

From the responsiveness point of view, we evaluate the amount of violation from the promised time (i.e., earliness and tardiness) and the number of OTIF services provided to the retailers relating to the proposed scenarios. When solving two scenarios with the case study data, we notice that no earliness happens in the network routes, i.e. both pickup and delivery routes. Moreover, we observed that the tardiness is observed only when dropping-off the orders at the retailer locations, i.e. in the delivery routes. Results reveal that the baseline scenario, involves 54 min of tardiness in delivery routes, while the optimized scenario, ends with 26 min of tardiness in delivery routes, contributing to 51.85% improvement in tardiness reduction. We find out that after solving two scenarios the baseline scenario results in providing OTIF services to 17

Table 4
Optimum results of the narrow time windows example.

Solution fronts	Routes Suppliers	Retailers	Solutions obtained via method(s)	OFV$_1$	OFV$_2$ (TE and TT)
1	CD-P2-P1-CD CD-P3-CD	CD-D3-D2-D1-CD	NWS-SWS-AEC	4286	480 (401 and 79)
2	CD-P3-P1-CD CD-P2-CD	CD-D3-D2-D1-CD	NWS-SWS-AEC	4293.2	438 (340 and 98)
3	CD-P3-P2-CD CD-P1-CD	CD-D3-D2-D1-CD	NWS-SWS-AEC	4340.4	414 (346 and 68)
4	CD-P3-P1-CD CD-P2-CD	CD-D3-D1-D2-CD	AEC	4534.4	372 (285 and 87)
5	CD-P3-P2-CD CD-P1-CD	CD-D3-D1-D2-CD	NWS-SWS-AEC	4581.6	348 (291 and 57)
6	CD-P1-P3-CD CD-P2-CD	CD-D1-D2-CD CD-D3-CD	AEC	5912.6	341 (311 and 30)
7	CD-P2-P1-CD CD-P3-CD	CD-D1-D2-CD CD-D3-CD	AEC	5959.8	317 (291 and 28)
8	CD-P3-P2-CD CD-P1-CD	CD-D1-D3-CD CD-D1-D2-CD	NWS -AEC	6275.6	294 (231 and 63)

Table 5
Optimum results of the wide time windows example.

Solution fronts	Routes Suppliers	Retailers	Solutions obtained via method(s)	OFV$_1$	OFV$_2$ (TE and TT)
1	CD-P2-P1-CD CD-P3-CD	CD-D3-D2-D1-CD	NWS-SWS-AEC	4286	401 (401 and 0)
2	CD-P3-P1-CD CD-P2-CD	CD-D3-D2-D1-CD	NWS-SWS-AEC	4293.2	340 (340 and 0)
3	CD-P3-P1-CD CD-P2-CD	CD-D2-D1-D3-CD	AEC	4328.4	321 (321 and 0)
4	CD-P1-P3-P2-CD CD-P1-CD	CD-D3-D2-D1-CD	NWS-SWS-AEC	4382.8	269 (269 and 0)
5	CD-P1-P3-P2-CD CD-P1-CD	CD-D2-D1-D3-CD	SWS-AEC	4418	250 (247 and 3)
6	CD-P1-P3-CD CD-P2-CD	CD-D1-D3-D2-CD	AEC	4560.4	199 (199 and 0)
7	CD-P1-P2-CD CD-P3-CD	CD-D1-D3-D2-CD	AEC	4623.6	189 (189 and 0)
8	CD-P1-P3-P2-CD CD-P1-CD	CD-D1-D3-D2-CD	NWS-SWS-AEC	4650	128 (128 and 0)

out of 20 retailers, while the optimized scenario can provide OTIF services to 18 out of 20 retailers.

Overall, we can conclude that recognizing order splitting in the routing problem improves both cost efficiency and customer responsiveness criteria, simultaneously. More precisely, including the latter has a greater impact on service level and OTIF indicators than cost efficiency. Therefore, it is recommended to consider the order splitting, when responsiveness and customer satisfaction is of paramount importance for the researchers and practitioners.

Table 6
Input data for pickup and delivery operations.

Type of service	Pickup operations	Delivery operations
Fleet input parameters	Medium/heavy-duty truck	Light-duty truck
Truck capacity	42 pallets (9-ton)	18 pallets (4-ton)
Number of trucks	4	10
Transportation cost ($/km)	0.87	1.18
Operational cost ($/node)	5	13
Service time (hour per vehicle per node)	3	1

Table 7
Comparative analysis of results.

Solutions	Baseline scenario	Optimized scenario	Improvement
Total cost ($)	9988.78	9834.82	1.54%
Transportation cost ($)	9838.78	9684.82	1.56%
Operational cost ($)	150	150	–
Truck capacity utilization[1]			
Medium/heavy-duty truck	87.50%	87.50%	–
Light-duty truck	65.46%	66.39%	1.40%
Travelled distance (km)	8811.30	8660.13	1.72%
Pickup route	1801.76	1722.99	4.37%
Delivery route	7009.50	6937.13	1.03%
Violation from promised time (min)	54	26	51.85%
Number of OTIF (out of 20)	17	18	5.56%

[1] Let $\widehat{y}_i^m$ be the optimal value for determining whether vehicle m is chosen to visit node j. We denote the optimal value of pickup and delivery amount of product type r by vehicle m by $\widehat{a}_{ir}^m, i \in \mathscr{P}$ and $\widehat{z}_{ir}^m, i \in \mathscr{D}$, respectively. Considering that, the truckload capacity utilization rate for pickup and delivery operations are calculated by $\frac{\sum_{r\in\mathscr{R}} f_r \widehat{a}_{ir}^m \widehat{y}_i^m}{Q_m} \times 100\%$ and $\frac{\sum_{i\in\mathscr{D}}\sum_{r\in\mathscr{R}} f_r \widehat{z}_{ir}^m \widehat{y}_i^m}{Q_m} \times 100\%$. Note that 100% load capacity utilization outbound, 0% on the return trip are excluded.

6. Conclusion

In this paper, we addressed a vehicle routing problem with cross-docking (VRPCD), in which splitting service in the pickup and delivery routes are allowed and every node has a time-windows to receive services. We formulated a bi-objective mixed-integer linear programming (MILP) model that minimizes (1) the transportation cost of routes and the operational cost of vehicles, and (2) the total amount of violation from the allowable interval (i.e., time windows) of each node. For solving the problem, we proposed a multi-objective heuristic algorithm and test several instances to verify the quality of Pareto front solutions. Results indicated that the proposed solution algorithm is superior, in terms of multi-objective performance metrics studied in this paper, to the counterpart algorithms available in the literature. Besides, we tested our methodology with case study data and conclude that the proposed model, which encourages multiple-visit, outperforms the baseline model in both cost efficiency and responsiveness objectives. In a nutshell, results suggested that simultaneous optimization of these conflicting objectives allowed the managers and decision-makers to achieve an appropriate supply chain competitive strategy with a certain level of profitability and customer satisfaction level. Our investigation also showed that taking into account splitting pickup and delivery could contribute to an improvement in OTIF indicators in the retail sector. The proposed model could shorten the total traveled distance and thus reduced transportation and operational cost.

The following features for further investigation are suggested. Determining an optimum number of operating dock-doors and truck scheduling decisions over a planning horizon would an interesting direction to develop this problem, which integrates a design decision with a multi-period inbound and outbound flow of trucks (Enderer, Contardo, & Contreras, 2017). An Open vehicle routing problem (OVRP), with cross-docking, can also be suggested as an extension to our paper. In OVRP, the vehicle may not return to the initial depot after pickup or delivery service to the last node (Yu, Jewpanya, & Perwira Redi, 2016). Another extension of this paper can be the consideration of its applicability in container shipping and maritime transportation. In the ship routing problem, time windows and harbor capacity constraints have a remarkable impact on pickup and delivery operations, and the inclusion of multiple-visits and splitting container shipping would be helpful to meet the cost efficiency and responsiveness objectives (Song & Dong, 2012).

CRediT authorship contribution statement

Asefeh Hasani Goodarzi: Conceptualization, Writing - original draft, Software. **Reza Tavakkoli-Moghaddam:** Supervision, Writing - review & editing. **Mehdi Amiri-Aref:** Investigation, Visualization. **Alireza Amini:** Resources, Project administration.

Table A1
Parameter tuning.

		Parameter limit values		Corresponding coded values		Tuned parameter value	Tuned coded value
		π_i^L	π_i^U	$\bar{\xi}_i$	ξi	$\pi_i^*(\omega)$	$\xi_i^*(\omega)$
π_1	S	100	300	−1	+1	193	−0.07
	L	200	400	−1	+1	327	0.27
π_2	S	4	8	−1	+1	5	−0.50
	L	6	12	−1	+1	9	0.00
π_3	S	0.2	0.8	−1	+1	0.55	0.17
	L	0.3	0.9	−1	+1	0.55	−0.17
π_4	S	0.2	0.8	−1	+1	0.60	0.33
	L	0.3	0.9	−1	+1	0.60	0.00
π_5	S	0.01	0.20	−1	+1	0.12	0.16
	L	0.05	0.25	−1	+1	0.12	−0.30

Appendix A

A.1. Parameter calibration

The efficiency of an algorithm is dramatically influenced by numerous input parameters and it is almost intractable to identify and to control the contributions of each input parameter. To overcome this difficulty, a response surface methodology (RSM) as the most commonly used multivariate statistic technique in analytical optimization, developed by Myers and Montgomery (1995) is applied in parameter tuning. The RSM collects the experimental data, which describes the behavior of a data set in response to any changes in the parameters used in an algorithm. The objective of the RSM is to simultaneously tune the input parameters in a way that high-quality solutions with less computational effort are yield. To obtain better solutions from the proposed algorithms, parameter tuning is implemented in Design-Expert software. After identifying the parameters, which statistically have a significant effect on the output, lower and upper limit values are feed into the RSM application of the Design-Expert software and it returns the best tune of the parameters. Algorithm A1 shows the parameter calibration method.

We identify a number of the initial population, several dominant solutions, percentages of crossover, percentages of absorption, and percentages of revolution as independent input parameters with a significant effect on the output of the algorithm and indicated them by $\pi = \{\pi_1, \pi_2, \pi_3, \pi_4, \pi_5\}$. For each independent input parameter, a lower and an upper limit value of the observations are given, denoted by π_i^L and π_i^U, respectively, and every value between these two limits is considered as possible input parameters (or observations). Then, the RSM converts the given values (or observations) to the coded parameters, indicated by $\xi_i(\omega)$, and evaluates the performance of the algorithm considering all possible combinations of the coded parameters corresponding to each independent parameter (or observations). Finally, it returns the best configuration of the input parameters to achieve the desirable and high-quality solutions with less computational efforts, where $\pi_i^*(\omega)$ and $\xi_i^*(\omega)$ represent the tuned parameter value and tuned coded value, respectively, as given in Table A1. Before feeding the data into the RSM application of the Design-Expert software, the following steps should be taken.

A.2. Revolution policies

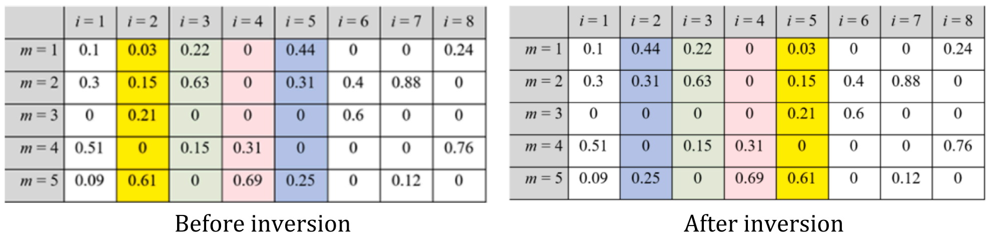
Three operators including inversion, swap, and insertion are used in the algorithm. In the inversion operator, two columns of the solution matrix are randomly chosen and interchanged. An illustrative example of an inversion operator is given in Fig. A1, in which nodes 2 and 5 are randomly selected for revolution mechanism. Inversion operation creates the matrix shown in Fig. A1(b), in which the 2nd and 5th columns are exchanged.

Algorithm A1. (*Parameter calibration method*)

```
SHORTLIST impactful parameters $\pi_1, \pi_2, \pi_3, \pi_4, \pi_5$
  For $\pi_i, i = 1, \cdots, 5$, do
    Generate $\Omega$ observations, $\pi_i(\omega), \omega = 1, \cdots, \Omega$
    $\pi_i^L = \min_{\omega=1,\cdots,\Omega}\{\pi_i(\omega)\}$ and $\pi_i^U = \max_{\omega=1,\cdots,\Omega}\{\pi_i(\omega)\}$
    $\xi_i(\omega) = \frac{\pi_i(\omega) - (\pi_i^U + \pi_i^L)/2}{(\pi_i^U - \pi_i^L)/2}$   Calculate $\bar{\xi}_i = \max_{\omega=1,\cdots,\Omega}\{\xi_i(\omega)\}$ and $\xi i = \min_{\omega=1,\cdots,\Omega}\{\xi_i(\omega)\}$
  FEED into RSM application of the Design-Expert software.
  OUTPUT: best value obtained from the observations $\pi_i^*(\omega)$ and the corresponding coding value $\xi_i^*(\omega)$.
```

In the swap operator, two columns of a solution matrix are randomly selected and the column permutation is applied. Note that column permutation here refers to a reversed order of the selected columns. Fig. A2 represents an example when the swap is applied, in which columns 2–5 are selected. After applying the swap operation, the outcome is shown in Fig. A2(b), in which the order of the aforementioned columns in the given matrix is reversed.

In the insertion operator, a column of the solution matrix is chosen randomly and its cells are replaced by new random numbers generated in the interval [0,1). An insertion operator is represented in Fig. A3, in which column 5 is chosen by chance, and its values are replaced by newly generated numbers.



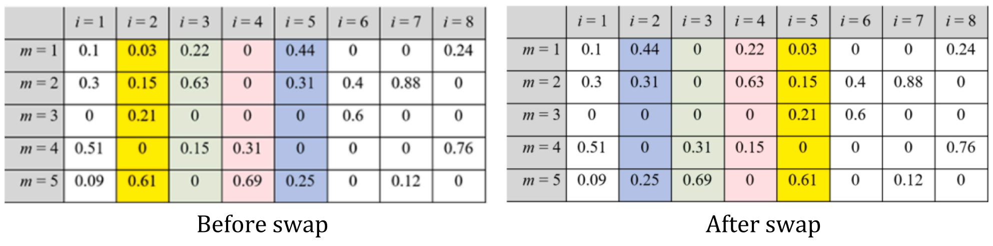
Before inversion

	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$	$i=6$	$i=7$	$i=8$
$m=1$	0.1	0.03	0.22	0	0.44	0	0	0.24
$m=2$	0.3	0.15	0.63	0	0.31	0.4	0.88	0
$m=3$	0	0.21	0	0	0	0.6	0	0
$m=4$	0.51	0	0.15	0.31	0	0	0	0.76
$m=5$	0.09	0.61	0	0.69	0.25	0	0.12	0

After inversion

	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$	$i=6$	$i=7$	$i=8$
$m=1$	0.1	0.44	0.22	0	0.03	0	0	0.24
$m=2$	0.3	0.31	0.63	0	0.15	0.4	0.88	0
$m=3$	0	0	0	0	0.21	0.6	0	0
$m=4$	0.51	0	0.15	0.31	0	0	0	0.76
$m=5$	0.09	0.25	0	0.69	0.61	0	0.12	0

Fig. A1. Inversion operator.



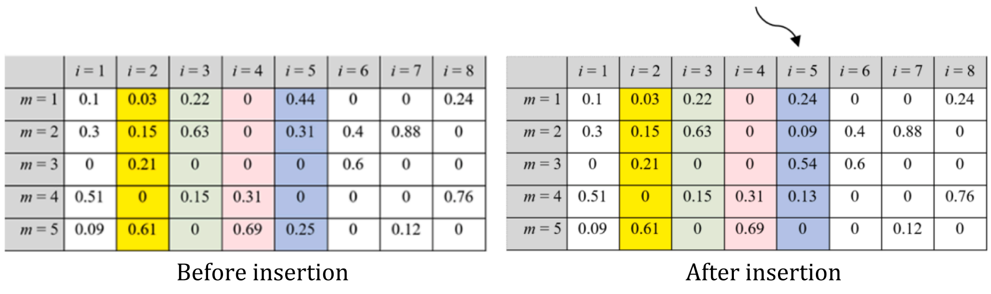
Before swap

	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$	$i=6$	$i=7$	$i=8$
$m=1$	0.1	0.03	0.22	0	0.44	0	0	0.24
$m=2$	0.3	0.15	0.63	0	0.31	0.4	0.88	0
$m=3$	0	0.21	0	0	0	0.6	0	0
$m=4$	0.51	0	0.15	0.31	0	0	0	0.76
$m=5$	0.09	0.61	0	0.69	0.25	0	0.12	0

After swap

	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$	$i=6$	$i=7$	$i=8$
$m=1$	0.1	0.44	0	0.22	0.03	0	0	0.24
$m=2$	0.3	0.31	0	0.63	0.15	0.4	0.88	0
$m=3$	0	0	0	0	0.21	0.6	0	0
$m=4$	0.51	0	0.31	0.15	0	0	0	0.76
$m=5$	0.09	0.25	0.69	0	0.61	0	0.12	0

Fig. A2. Swap operator.



Before insertion

	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$	$i=6$	$i=7$	$i=8$
$m=1$	0.1	0.03	0.22	0	0.44	0	0	0.24
$m=2$	0.3	0.15	0.63	0	0.31	0.4	0.88	0
$m=3$	0	0.21	0	0	0	0.6	0	0
$m=4$	0.51	0	0.15	0.31	0	0	0	0.76
$m=5$	0.09	0.61	0	0.69	0.25	0	0.12	0

After insertion

	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$	$i=6$	$i=7$	$i=8$
$m=1$	0.1	0.03	0.22	0	0.24	0	0	0.24
$m=2$	0.3	0.15	0.63	0	0.09	0.4	0.88	0
$m=3$	0	0.21	0	0	0.54	0.6	0	0
$m=4$	0.51	0	0.15	0.31	0.13	0	0	0.76
$m=5$	0.09	0.61	0	0.69	0	0	0.12	0

Fig. A3. Insertion operator.

Appendix B. Supplementary material

Supplementary data to this article can be found online at https://doi.org/10.1016/j.cie.2020.106832.

References

Ahkamiraad, A., & Wang, Y. (2018). Capacitated and multiple cross-docked vehicle routing problem with pickup, delivery, and time windows. *Computers and Industrial Engineering, 119*, 76–84.

Alpan, G., Larbi, R., & Penz, B. (2011). A bounded dynamic programming approach to schedule operations in a cross docking platform. *Computers and Industrial Engineering, 60*(3), 385–396.

Amini, A., & Tavakkoli-Moghaddam, R. (2016). A bi-objective truck scheduling problem in a cross-docking center with probability of breakdown for trucks. *Computers and Industrial Engineering, 96*, 180–191.

Archetti, C., Savelsbergh, M., & Speranza, M. G. (2008). To split or not to split: That is the question. *Transportation Research Part E: Logistics and Transportation Review, 44*(1), 114–123.

Atashpas-Gargari, E., & Lucas, C. (2007). Imperialist competitive algorithm: An algorithm for optimization inspired by imperialist competitive. In: Proceeding IEEE Congress on Evolutionary computation. CEC 2007, Singapore.

Baniamerian, A., Bashiri, M., & Tavakkoli-Moghaddam, R. (2019). Modified variable neighborhood search and genetic algorithm for profitable heterogeneous vehicle routing problem with cross-docking. *Applied Soft Computing, 75*, 441–460.

Baños, R., Ortega, J., Gil, C., Márquez, A. L., & Toro, F. (2013). A hybrid meta-heuristic for multi-objective vehicle routing problems with time windows. *Computers and Industrial Engineering, 65*(2), 286–296.

Battara, M., Cordeau, J. F., & Iori, M. (2014). Pickup-and-delivery problems for goods transportation. In P. Toth, & D. Vigo (Eds.), *Vehicle routing: MOSSIAM Series on Optimization - Society for Industrial and Applied Mathematics*.

Bin, J. (2006). Cross-docking. M.Sc. Thesis, Technical University of Denmark, DTU.

Boloori Arabani, A. R., Fatemi Ghomi, S. M. T., & Zandieh, M. (2011). Meta-heuristics implementation for scheduling of trucks in a cross-docking system with temporary storage. *Expert Systems with Applications, 38*(3), 1964–1979.

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

Chen, H. K., Chou, H. W., Hsueh, C. F., & Yen-Ju, Y. (2015). The paired many-to-many pickup and delivery problem: An application. *TOP, 23*, 220–243.

Chen, M.-C., Hsiao, Y.-H., Reddy, R. H., & Tiwari, M. K. (2016). The Self-Learning Particle Swarm Optimization approach for routing pickup and delivery of multiple products with material handling in multiple cross-docks. *Transportation Research Part E: Logistics and Transportation Review, 91*(1), 208–222.

Chen, F., & Lee, C.-Y. (2009). Minimizing the makespan in a two-machine crossdocking flow shop problem. *European Journal of Operational Research, 193*(1), 59–72.

Chen, Q. F., Li, K. P., & Liu, Z. X. (2014). Model and algorithm for an unpaired pickup and delivery vehicle routing problem with split loads. *Transportation Research Part E: Logistics and Transportation Review, 69*, 218–235.

Chen, F., & Song, K. (2009). Minimizing makespan in two-stage hybrid cross docking scheduling problem. *Computers and Operations Research, 36*(6), 2066–2073.

Czyzzak, P., & Jaszkiewicz, A. (1998). Pareto simulated annealing– a metaheuristic technique for multiple-objective combinatorial optimization. *Journal of Multi-Criteria Decision Analysis, 7*(1), 34–47.

Dantzig, G. B., & Ramser, J. H. (1959). The truck dispatching problem. *Management Science, 6*, 80–91.

Deb, K., Agrawal, S., Pratap, A., & Meyarivan, T. (2000). A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In M. Schoenauer, et al. (Eds.) Parallel Problem Solving from Nature PPSN VI. PPSN 2000. Lecture Notes in Computer Science, 1917. Berlin, Heidelberg: Springer.

Deb, K. (1999). Multi-objective genetic algorithms: Problem difficulties and construction of test problems. *Evolutionary Computation, 7*(3), 205–230.

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multi-objective genetic algorithm: NSGA-II. *IEEE Transaction on Evolutionary Computation, 6*(2), 182–197.

Dror, M., Laporte, G., & Trudeau, P. (1994). Vehicle routing with split deliveries. *Discrete Applied Mathematics, 50*(3), 239–254.

Dror, M., & Trudeau, P. (1989). Savings by split delivery routing. *Transportation Science, 23*, 141–145.
Dondo, R., & Cerdá, J. (2013). A sweep-heuristic based formulation for the vehicle routing problem with cross-docking. *Computers and Chemical Engineering, 48*, 293–311.
Dulebenets, M. A. (2019). A Delayed Start Parallel Evolutionary Algorithm for just-in-time truck scheduling at a cross-docking facility. *International Journal of Production Economics, 212*, 236–258.
Enderer, F., Contardo, C., & Contreras, I. (2017). Integrating dock-door assignment and vehicle routing with cross-docking. *Computers and Operations Research, 88*, 30–43.
García-Nájera, A., Bullinaria, J. A., & Gutiérrez-Andrade, M. A. (2015). An evolutionary approach for multi-objective vehicle routing problems with backhauls. *Computers and Industrial Engineering, 81*, 90–108.
Ghannadpour, S. F., Noori, S., & Tavakkoli-Moghaddam, R. (2014). A multi-objective vehicle routing and scheduling problem with uncertainty in customers' request and priority. *Journal of Combinatorial Optimization, 28*(2), 414–446.
Goldberg, D. E. (1989). *Genetic algorithms in search, optimization and machine learning* (1st ed.). Boston, MA, USA: Addison-Wesley Longman Publishing Co. Inc.
Grimault, A., Bostel, N., & Lehuédé, F. (2017). An adaptive large neighborhood search for the full vehicle load pickup and delivery problem with resource synchronization. *Computers and Operations Research, 88*, 1–14.
Han, A. F. W., & Chu, Y. C. (2016). A multi-start heuristic approach for the split-delivery vehicle routing problem with minimum delivery amounts. *Transportation Research Part E: Logistics and Transportation Review, 88*, 11–31.
Hasani Goodarzi, A., & Zegordi, S. H. (2016). A location- routing problem for cross-docking networks: A biogeography-based optimization algorithm. *Computers and Industrial Engineering, 102*, 132–146.
Hassanzadeh, A., & Rasti-Barzoki, M. (2017). Minimizing total resource consumption and total tardiness penalty in a resource allocation supply chain scheduling and vehicle routing problem. *Applied Soft Computing, 58*, 307–323.
Healy, P., & Moll, R. (1995). A new extension of local search applied to the Dial-A-Ride Problem. *European Journal of Operational Research, 83*(1), 83–104.
Javanmard, S., Vahdani, B., & Tavakkoli-Moghaddam, R. (2014). Solving a multi-product distribution planning problem in cross docking networks: An imperialist competitive algorithm. *The International Journal of Advanced Manufacturing Technology, 70*(9), 1709–1720.
Jiang, J., Ng, K. M., Poh, K. L., & Teo, K. M. (2014). Vehicle routing problem with a heterogeneous fleet and time windows. *Expert Systems with Applications, 41*(8), 3748–3760.
Knowles, J.D., & Corne, D.W. (1999). The Pareto archived evolution strategy: A new baseline algorithm for Pareto multi-objective optimization. In Congress on Evolutionary Computation (CEC99) (Vol. 1, pp. 98–105). Washington, DC, USA.
Kourank Beheshti, A., Hejazi, S. R., & Alinaghian, M. (2015). The vehicle routing problem with multiple prioritized time windows: A case study. *Computers and Industrial Engineering, 90*, 402–413.
Kumar, R. S., Kondapaneni, K., Dixit, V., Goswami, A., Thakur, L. S., & Tiwari, M. K. (2016). Multi-objective modeling of production and pollution routing problem with time window: A self-learning particle swarm optimization approach. *Computers and Industrial Engineering, 99*, 29–40.
Ladier, A. L., & Alpan, G. (2016). Cross-docking operations: Current research versus industry practice. *Omega, 62*, 145–162.
Lee, Y. H., Jung, J. W., & Lee, K. M. (2006). Vehicle routing scheduling for cross-docking in the supply chain. *Computers and Industrial Engineering, 51*, 247–256.
Lehuédé, F., Péton, O., & Tang, X. (2014). Optimization of customer orders routing in a collaborative distribution network. Rapport interne 14-6-AUTO.
Liao, C.-J., Lin, Y., & Shih, S. C. (2010). Vehicle routing with cross-docking in the supply chain. *Expert Systems with Applications, 37*(10), 6868–6873.
Liman, S., & Ramaswamy, S. (1994). Earliness/tardiness scheduling problems with a common delivery window. *Operations Research Letters, 15*, 195–203.
Liu, R., Jiang, Z., Fung, R., Chen, F., & Liu, X. (2010). Two-phase heuristic algorithms for full vehicle loads multi-depot capacitated vehicle routing problem in carrier collaboration. *Computers and Operations Research, 37*(5), 950–959.
Lwin, K., Qu, R., & Kendall, G. (2014). A learning-guided multi-objective evolutionary algorithm for constrained portfolio optimization. *Applied Soft Computing, 24*(1), 757–772.
Luo, H., Yang, X., & Wang, K. (2019). Synchronized scheduling of make to order plant and cross-docking warehouse. *Computers and Industrial Engineering, 138*, Article 106108.
Maknoon, Y., & Laporte, G. (2017). Vehicle routing with cross-dock selection. *Computers and Operations Research, 77*, 254–266.
Marseguerra, M., Zio, E., & Podofillini, L. (2002). Condition-based maintenance optimization by means of genetic algorithms and Monte Carlo simulation. *Reliability Engineering and System Safety, 77*(2), 151–166.
Matzler, K., Veider, V., & Kathan, W. (2015). Adapting to the sharing economy. *MIT Sloan Management Review, 56*(2), 71–77.
Melián-Batista, B., Santiago, A., Angel-Bello, F., & Alvarez, A. (2014). A bi-objective vehicle routing problem with time windows: A real case in Tenerife. *Applied Soft Computing, 17*, 140–152.
Mohammadi, M., Jolai, F., & Tavakkoli-Moghaddam, R. (2013). Solving a new stochastic multi-mode p-hub covering location problem considering risk by a novel multi-objective algorithm. *Applied Mathematical Modelling, 37*(24), 10053–10073.
Mohtashami, A., Tavanab, M., Santos-Arteagad, F. J., & Fallahian-Najafabadi, A. (2015). A novel multi-objective meta-heuristic model for solving cross-docking scheduling problems. *Applied Soft Computing, 31*, 30–47.
Mollanoori, H., Tavakkoli-Moghaddam, R., Triki, C., Hajiaghaei-Keshteli, M., & Sabouhi, F. (2019). Extending the solid step fixed-charge transportation problem to consider two-stage networks and multi-item shipments. *Computers and Industrial Engineering, 137*, Article 106008.
Mousavi, S. M., Tavakkoli-Moghaddam, R., & Jolai, F. (2013). A possibilistic programming approach for the location problem of multiple cross-docks and vehicle routing scheduling under uncertainty. *Engineering Optimization, 45*(10), 1223–1249.
Musa, R., Arnaout, J.-P., & Jung, H. (2010). Ant colony optimization algorithm to solve for the transportation problem of cross-docking network. *Computers and Industrial Engineering, 59*(1), 85–92.
Myers, R. H., & Montgomery, D. C. (1995). *Response surface methodology: Process and product optimization using designed experiments*. New York: John Wiley and Sons.
Neves-Moreira, F., Amorim, P., Guimares, L., & Almada-Lobo, B. (2016). A longhaul freight transportation problem: Synchronizing resources to deliver requests passing through multiple transshipment locations. *European Journal of Operational Research, 248*(2), 487–506.
Rabbani, M., Heidari, R., Farrokhi-Asl, H., & Rahimi, N. (2018). Using metaheuristic algorithms to solve a multi-objective industrial hazardous waste location-routing problem considering incompatible waste types. *Journal of Cleaner Production, 170*, 227–241.
Santos, F. A., Mateus, G. R., & da Cunha, A. S. (2013). The pickup and delivery problem with cross-docking. *Computers and Operations Research, 40*, 1085–1093.
Savelsbergh, M., & Woensel, T. V. (2016). City logistics: Challenges and opportunities. *Transportation Science, 50*(2), 579–590.
Shahmardan, A., & Sajadieh, M. S. (2020). Truck scheduling in a multi-door cross-docking center with partial unloading–Reinforcement learning-based simulated annealing approaches. *Computers and Industrial Engineering, 139*, Article 106134.
Sidney, J. (1977). Optimal single-machine scheduling with earliness and tardiness penalties. *Operations Research, 25*(1), 62–69.
Sierra M.R., & Coello Coello C.A. (2005). Improving PSO-based multi-objective optimization using crowding, mutation and ∊-dominance. In C.A., Coello Coello, A. Hernández Aguirre, E. Zitzler (Eds.) Evolutionary Multi-Criterion Optimization. EMO 2005. Lecture Notes in Computer Science, 3410. Berlin, Heidelberg: Springer.
Silva, M. M., Subramanian, A., & Ochi, L. S. (2015). An iterated local search heuristic for the split delivery vehicle routing problem. *Computers and Operations Research, 53*, 234–249.
Song, D. P., & Dong, J. X. (2012). Cargo routing and empty container repositioning in multiple shipping service routes. *Transportation Research Part B: Methodological, 46*(10), 1556–1575.
Tavakkoli-Moghaddam, R., Azarkish, M., & Sadeghnejad, A. (2011). A new hybrid multi objective Pareto archive PSO algorithm for a bi-objective job shop scheduling problem. *Expert System with Applications, 38*, 10812–10821.
Vafaeenezhad, T., Tavakkoli-Moghaddam, R., & Cheikhorouhou, N. (2019). Multi-objective mathematical modelling for sustainable supply chain management in the paper industry. *Computers and Industrial Engineering, 135*, 1092–1102.
Vahdani, B., Tavakkoli-Moghaddam, R., Zandieh, M., & Razmi, J. (2012). Vehicle routing scheduling using an enhanced hybrid optimization approach. *Journal of Intelligent Manufacturing, 23*, 759–774.
Van Veldhuizen, D. A., & Lamont, G. B. (1998). Multiobjective evolutionary algorithm research: a history and analysis. Technical Report, https://mitpress.mit.edu.
Wang, J., Jagannathan, A. K. R., Zuo, X., & Murray, C. C. (2017). Two-layer simulated annealing and tabu search heuristics for a vehicle routing problem with cross docks and split deliveries. *Computers and Industrial Engineering, 112*, 84–98.
Wen, M., Larsen, J., Clausen, J., Cordeau, J.-F., & Laporte, G. (2009). Vehicle routing with cross-docking. *Journal of Operations Research Society, 60*(12), 1708–1718.
Xu, D., Li, K., Zou, X., & Liu, L. (2017). An unpaired pickup and delivery vehicle routing problem with multi-visit. *Transportation Research Part E: Logistics and Transportation Review, 103*, 218–247.
Yin, P.-Y., & Chuang, Y.-L. (2016). Adaptive memory artificial bee colony algorithm for green vehicle routing with cross-docking. *Applied Mathematical Modelling, 40*(21–22), 9302–9315.
Yu, V. F., Jewpanya, P., & Perwira Redi, A. A. N. (2016). Open vehicle routing problem with cross-docking. *Computers and Industrial Engineering, 94*, 6–17.
Zitzler, E., Deb, K., & Thiele, L. (2000). Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary computation, 8*(2), 173–195.
Zitzler, E., & Thiele, L. (1999). Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation, 3*(4), 257–271.