JOURNAL OF INDUSTRIAL AND
MANAGEMENT OPTIMIZATION

doi:10.3934/jimo.2021002

A MODEL AND TWO HEURISTIC METHODS FOR THE MULTI-PRODUCT INVENTORY-LOCATION-ROUTING PROBLEM WITH HETEROGENEOUS FLEET

Ömer Arslan*

Department of Industrial Engineering
Gazi University
Ankara, Turkey

Selçuk Kürşat İşleyen

Department of Industrial Engineering
Gazi University
Ankara, Turkey

(Communicated by Harish Garg)

Abstract. The Multi-Product Inventory-Location-Routing Problem with heterogeneous fleet considers a supply chain, which comprises multiple producers, potential distribution centers (DCs) with opening capacity levels and geographically scattered retailers each of which has deterministic demand over a discrete planning horizon. The goal is determining a set of DCs with their capacity levels to open, assigning retailers to the opened DCs, finding product quantities to be ordered by and distributed from opened DCs and determining the fleet and routes to satisfy the demands of retailers with minimum cost. A mixed-integer linear programming model is proposed to describe the problem, which is strengthened by two valid inequalities. Since the commercial solver can only solve the very small-sized instances within a reasonable time, two heuristic methods are developed. Results show that the proposed valid inequalities are effective and both methods provide important savings in acceptable run times compared to the commercial solver.

1. **Introduction.** According to a study by Blanchard [2], in the USA, more than $1 trillion a year is spent on the optimization of supply chains. To make the relevant optimizations, two main processes, namely production and logistics are generally reviewed. The production process involves mainly the production planning, the handling of materials and the reorganization of the facilities. The logistics process involves the facility location, the procurement of raw materials, the distribution of goods to customers and inventory management issues.

According to a 2005 study by MacroSy research and technology organization, the total cost of the logistics for year in the USA in 2002 was nearly $910 billion, which constituted 8.7% of the country's gross domestic product. Of the $910 billion, $577 billion was for transportation costs, and $298 billion was for inventory holding and

2020 *Mathematics Subject Classification.* Primary: 90B05, 90B06, 90C11; Secondary: 90C27.
Key words and phrases. Heuristic, inventory-location-routing problem, location-routing problem, logistics, supply chain.

* Corresponding author: omer.arslan1@gazi.edu.tr.

the operating costs of distribution centers. The supply chains of the organizations should be designed and optimized in a way that reduces these costs.

Three fundamental problems must be addressed when designing and optimizing supply chains: location-allocation, on the strategic level; inventory management, on the tactical level; and vehicle routing problems, on the operational level. Location-allocation problems are related to choosing a subset of facilities (distribution centers and depots etc.) from candidate facilities and allocating retailers/customers to these facilities in order to minimize the total of facility opening and operating costs. Inventory management problems determine the amount of product(s) for supply and distribution, for each period, in order to minimize the inventory holding and stockout costs. Vehicle routing problems are related to finding the best vehicle fleet and routes in order to minimize fleet and distribution costs.

In many of the previous studies on the aforementioned problems, researchers have studied the problems on their own and in pairs. On the one hand, some researchers claim that the problems should not be addressed simultaneously, as strategic decisions are fixed for long periods of time (more than a year), while tactical and operational decisions are changeable within shorter periods of time. On the other hand, as noted by Shen and Qi [18]; "These three problems of a supply chain are highly related. An effective distribution scheme depends on the locations of facilities. The allocation of retailers to open distribution centers significantly impacts the optimal order sizes placed at the open distribution centers, as well as each center's routing system", most of the researchers including us claim that the problems should be addressed simultaneously in order to find optimal configurations. Javid and Azad [10] realized this recommendation first and named the considered problem set as the "Inventory-Location-Routing Problem" (ILRP). This problem is one of deciding the location of distribution centers while considering routing and inventory policies over a finite planning horizon.

When we look at the application areas of ILRP, we see medicine/food delivery in humanitarian logistics, ammunition, and other supplies distribution in military logistics, supermarket logistics and many of the logistics cases in which leasing distribution centers are common and distribution to the customers are made in different frequencies. In the long run, when the location costs are in the same order of magnitude as operational costs, as in the pharmaceutical industry, the ILRP comes into play. When direct deliveries might lead to suboptimal solutions, as in the case when vehicle capacity exceeds the maximum demand largely, the ILRP should be considered. We must also use ILRP to perform "what if" analysis during the assessment of supply chain design.

Suppose we think about the case of a catastrophe (earthquake, tsunami and others) where humanitarian logistics must be considered to distribute water, medicine, and other supplies using limited resources. In that case, we face ILRP to locate the facilities where these supplies will be stored/distributed from and determine how to perform distribution under capacity constraints. Failing to provide necessary supplies in time during these situations could cause loss of lives or permanent damages to people's health. Another crucial reason to study ILRP is the increasing importance of greener logistics. To decrease the harm given to the environment and decrease the energy consumed, we should open facilities in convenient places, maintain inventories at these facilities in sufficient levels and minimize the road lengths passed during the distribution process. After making necessary optimizations, we reduce the energy consumed for product storage in

cold surroundings and the energy consumed for product distribution. Besides, less carbon dioxide/heat is emitted to the atmosphere.

Regarding the studies on ILRP, Table 1 summarizes the main literature review. The table does not cover the studies regarding fuzzy demand or periodic problems. Columns "# of Pe." and "# of Pr." denote the number of periods and the number of products.

TABLE 1. Main literature review summary on ILRP

Study	**# of Pe.**	**# of Pr.**	**Demand**	**Fleet**	**Solution method**
Liu and Lee (2003)	Single	Single	Stochastic	Homo.	Sequential heuristic
Liu and Lin (2005)	Single	Single	Stochastic	Homo.	Hybrid heuristic
Ambrosino and Scutella (2005)	Single	Single	Deterministic	Homo.	Com. solver
Ma and Davidrajuh (2005)	Single	Single	Stochastic	Homo.	Sequential heuristic
Shen and Qi (2007)	Single	Single	Stochastic	Homo.	Branch & Bound
Javid and Azad (2010)	Single	Single	Stochastic	Homo.	Tabu search & Sim.Annealing
Granada and Silva (2012)	Multiple	Single	Deterministic	Homo.	Column generation
Sajjadi et al. (2013)	Single	Multiple	Deterministic	Homo.	Sequential heuristic
Guerrero et al. (2013)	Multiple	Single	Deterministic	Homo.	Hybrid heuristic
Nekooghadirli et al. (2014)	Multiple	Multiple	Stochastic	Hete.	Multi-objective meta-heuristic methods
Zhang et al. (2014)	Multiple	Single	Deterministic	Homo.	Hybrid heuristic
Guerrero et al. (2015)	Multiple	Single	Deterministic	Homo.	Relax & Price
Hiassat et al. (2017)	Multiple	Single	Deterministic	Homo.	Genetic algorithm
Saragih et al. (2019)	Single	Single	Stochastic	Homo.	Sim.Annealing
Our study	Multiple	Multiple	Deterministic	Hete.	Sequential heuristic Hybrid heuristic

Liu and Lee [12] and Liu and Lin [11] considered a single-product, single-period, stochastic demand multi-depot location-routing problem. Liu and Lee [12] proposed a two-phase heuristic method to solve the problem. In the first phase, they find the initial solution through route-first, location-allocation second approach and then in the second phase, improve this solution via the heuristic method. On the other hand, Liu and Lin [11] separated the problem into two subproblems, location-allocation and inventory-routing. They proposed a hybrid heuristic method, which improves both sub problems' initial solutions using tabu search and simulated annealing.

Ambrosino and Scutella [1] proposed models representing different scenarios including plants, central depots, regional depots, transit points, clients and big clients relevant to distribution network design. They tried to solve a single-product, single period, deterministic demand ILRP instance including 13 depots, 95 retailers and 17 vehicles using CPLEX 7.0 within 25 h and reached lower bound with a 23.71% gap.

Ma and Davidrajuh [13] considered a single-product, single period, stochastic demand ILRP and proposed a two-stage iterative solution method. In the first stage, location-allocation decisions are found by solving a mixed integer model. In the second stage, they find inventory decisions using probability theory and routing decisions through a genetic algorithm.

Shen and Qi [18] formulated the single-product, single-period, stochastic demand ILRP as a nonlinear model and proposed a method using Lagrangian relaxation embedded in a branch and bound algorithm. Javid and Azad [10] improved Shen and Qi [18]'s study and presented a model to optimize location, allocation, capacity, inventory and routing decisions simultaneously for the first time. They cast the problem as a mixed integer convex program to solve small-sized instances with

branch and bound method. They also proposed a method, which is hybridization of tabu search and simulated annealing to solve medium and large problems.

Granada and Silva [6] considered single-product, multi-period, deterministic demand ILRP and proposed a column generation-based method by decomposing the problem into a master problem (location) and an auxiliary problem (inventory-routing). They solved small-sized problems with the proposed method, but could not solve big-sized problems because of insufficient RAM.

Sajjadi *et al.* [16] considered a multi-product, single period, deterministic demand ILRP and proposed a two-stage method. In the first stage, they find location-allocation and routing decisions. In the second stage, they take care of inventory management decisions. They solved instances including two products, ten depots, 200 customers nearly within 10 seconds.

Guerrero *et al.* [7] presented single-product, multi-period, deterministic demand ILRP and proposed two methods. The first method is sequential, but the second one is hybrid, which embeds an exact approach within a heuristic scheme. They achieved significant savings with hybrid heuristic, but they could only solve relatively small instances compared to the other studies.

Nekooghadirli *et al.* [14] considered bi-objective, multi-product, multi-period, stochastic demand and bi-objective ILRP. They tried to minimize the total cost and the maximum mean time spent on delivery. They proposed four multi-objective meta-heuristic methods.

Zhang *et al.* [19] proposed a model for single-product, multi-period, deterministic demand ILRP which makes cross-dock depots assumption. They proposed a hybrid method consisting of initialization, intensification and post-optimization stages. They solved instances including 25 depots, 300 customers and seven periods nearly within 40 minutes.

Guerrero *et al.* [8] considered the same problem studied by Guerrero *et al.* [7] and proposed a method: the hybridization of column generation, Lagrangian relaxation and local search. The proposed method founded solutions in shorter computational times than Guerrero *et al.* [7] study at the expense of solution quality. Ghorbani and Jokar [4] studied multi-product, multi-period ILRP that allows backlogging. They proposed a hybrid heuristic algorithm based on simulated annealing and imperialist competitive algorithm.

Hiassat *et al.* [9] addressed the multi-period ILRP for perishable products. They solved small-sized instances within acceptable times through CPLEX solver and proposed a genetic algorithm for medium and large-sized instances.

Finally, Saragih *et al.* [17] considered single-product, probabilistic demand, multi-period ILRP in a three-echelon supply chain. They proposed a heuristic method consisting of two stages. In the first stage, the initial solution is constructed and in the second stage, it is improved iteratively using simulated annealing.

Considering the studies on ILRP up to now, we see that ILRP is dealt with different assumptions in different studies. A comprehensive/generic approach, which takes care of many aspects of the problem altogether is needed. Therefore, in this study we consider multi-product ILRP (MILRP) with heterogeneous fleet, which simultaneously takes care of many aspects of the problem. Main contributions of this study are threefold: first, a mixed-integer linear programming model is proposed to formulate the considered problem. Second, a sequential heuristic method and third a hybrid heuristic method is developed to solve all sized instances of the MILRP with heterogeneous fleet.

The outline of the paper is as follows. In the next section, the mathematical model for MILRP with the heterogeneous fleet is presented first, and then two valid inequalities proposed for ILRP are adapted to the MILRP. In the third section, the sequential heuristic method and in the fourth section the hybrid heuristic method is presented in detail. We give the computational results in the fifth section. The last section includes the conclusions and future research.

2. **Problem definition.** The problem studied in this paper considers a supply chain design consisting of multiple producers, potential distribution centers (DCs) and a set of retailers for a finite-horizon with a limited number of periods. The goal is to determine a set of DCs with their capacity levels to open, assign retailers to the opened DCs and for all periods find the product quantities to be ordered by and distributed from opened DCs and determine the routes to satisfy the demands of retailers with minimum cost. Representation of the problem is given below:

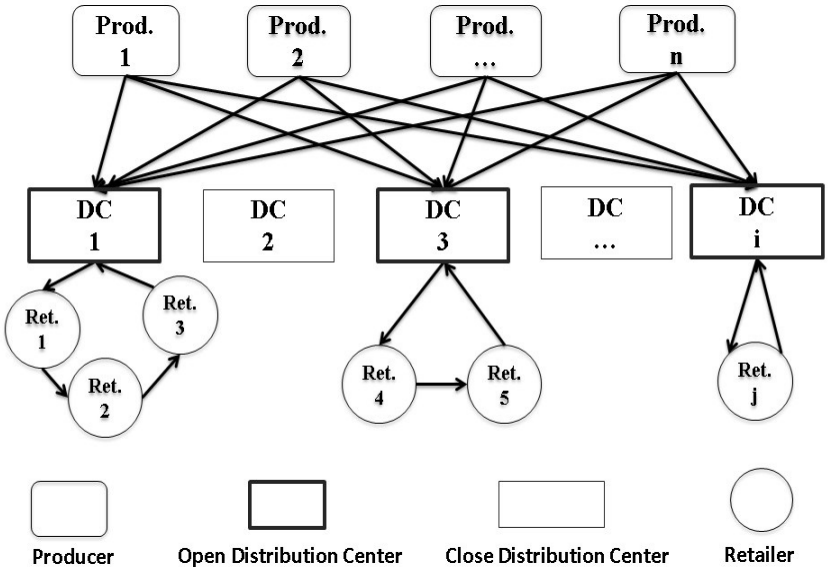


FIGURE 1. Representation of the problem

The assumptions of the problem are as follows. Each retailer faces a deterministic demand for each type of product for multiple periods. Each producer produces only a single product. Each DC can be opened in one of the capacity levels (small, medium and large) with different costs. Location and allocation decisions are strategic and fixed for all periods. Inventories are kept both at the DCs and at retailers. There can be initial inventories at the DCs and retailers. There is a holding plus obsolescence cost for each type of product kept at DCs and retailers. Backlogging or stock-out is not allowed. In any period, each vehicle travels at most on one route, and retailers are visited at most once. Vehicles depart from and return to the DCs. Product delivery from producers to DCs is dedicated, whereas from DCs to retailers is non-dedicated. Deliveries are made by heterogeneous vehicle fleet, which contains three types (small, medium and large) of vehicles. The distribution cost includes travelling distance related cost and vehicle fixed cost (using a vehicle

at least once cost). Each vehicle type has a different fixed cost and unit cost of travelling per km. After solving the problem, we will get the answers to these questions:

- Which DCs should be opened?
- What should be the capacity levels of each opened DC?
- Which retailer(s) should be allocated to each opened DC?
- How many vehicles from each type are required by opened DCs?
- For each period and each type of product;
  - In which amounts products should be ordered by opened DCs?
  - In which amounts opened DCs should deliver products to retailers?
- For each period, how should be the vehicle routes from opened DCs to retailers?

2.1. **Mathematical model.** The model for MILRP with heterogeneous fleet (M1 model), based on the ILRP model proposed by Guerrero *et al.* [7], is given below. Our formulation adds multi-product, heterogeneous fleet and capacity levels for DCs extra features. We remove w variables from the referenced model and introduce s, w, q variables to decrease the number of decision variables and make the model easily understandable. Guerrero *et al.* [7] separated w variables for first and second echelon with superscripts and used two subscripts for periods (A different point of view). However, we use only one subscript for periods and add a subscript for the product type. After these modifications, we also add inventory balance equations for DCs and retailers to correct the model. An analysis regarding this modification is presented in the Results section.

Sets

N : products,
I : potential DCs,
J : retailers,
$K = I \cup J$,
$A = \{(i,\ j)\,;i,\ j \in K,\ i \neq j\}$: arcs,
$G = (K,\ A)$: weighted and directed graph,
V : vehicles,
$B = \{1,2,3\}$: vehicle types,
$E = \{1,2,3\}$: capacity levels for DCs,
$H = \{1,2,..,p\}$: periods,
$H^0 = \{0\} \cup\ H\ = \{0,1,2,..,p\}$: periods including initial period,

Parameters

$dist_{ij}$: length of $i-j$ arc in km $(\forall\,(i,j) \in A)$,
c^b : unit cost of travelling per km for type b vehicle $(\forall b \in B)$,
c^b_{ij} : cost of travelling for $i-j$ arc passed by type b vehicle
$(\forall b \in B, \forall\,(i,j) \in A)\,, c^b_{ij} = c^b.\ dist_{ij}$
d_{jtn} : demand of retailer j for product n at period t $(\forall j \in J,\ \forall t \in H,\ \forall n \in N)$,
OC_{in} : ordering cost of product n for DC i $(\forall i \in I, \forall n \in N)$,
h_{ktn} : total unit holding with obsolescence cost of node k at period t
for product n $(\forall k \in K,\ \forall t \in H,\ \forall n \in N)$,

OPC_i^e : opening cost of DC i with capacity level $e\,(\forall i \in I,\ \forall e \in E)$,
MCQ_i : periodic operating cost quotient of DC $i\,(\forall i \in I)$,
W_i^e : capacity of DC i opened at capacity level $e\,(\forall i \in I,\ \forall e \in E)$,
W_j : capacity of retailer $j\,(\forall j \in J)$,
Q^b : capacity of type b vehicle $(\forall b \in B)$,
FC^b : cost of using type b vehicle at least once $(\forall b \in B)$,
IS_{kn} : initial inventory of node k for product $n\,(\forall k \in K,\ \forall n \in N)$,
α : weight factor for ordering cost,
β : weight factor for distribution cost,
θ : weight factor for inventory and obsolescence cost total for DCs,
γ : weight factor for inventory and obsolescence cost total for retailers.

Decision variables

$$y_i^e \ : \begin{cases} 1 & \text{if DC } i \text{ is opened at capacity level } e\ \ (\forall i \in I,\ \forall e \in E) \\ 0 & \text{otherwise} \end{cases}$$

$$f_{ij} \ : \begin{cases} 1 & \text{if retailer } j \text{ is assigned to DC } i\ \ (\forall i \in I,\ \forall j \in J) \\ 0 & \text{otherwise} \end{cases}$$

$$z_{itn} \ : \begin{cases} 1 & \text{if product } n \text{ is ordered by DC } i \text{ at period t} \\ & (\forall i \in I,\ \forall t \in H,\ \forall n \in N) \\ 0 & \text{otherwise} \end{cases}$$

$$x_{ijtv}^b \ : \begin{cases} 1 & \text{if the arc } (i,j) \text{ is crossed from } i \text{ to } j \text{ by vehicle } v \text{ of type } b \\ & \text{at period } t, (\forall i,j \in K,\ \forall t \in H,\ \forall b \in B,\ \forall v \in V) \\ 0 & \text{otherwise} \end{cases}$$

s_{itn} : quantity of product n ordered by DC i at period t $(\forall i \in I,\ \forall t \in H,\ \forall n \in N)$,
w_{ijtnv}^b : quantity of product n delivered from DC i to retailer j at period t by type b vehicle $(\forall i \in I,\ \forall j \in J,\ \forall t \in H,\ \forall n \in N,\ \forall b \in B,\ \forall v \in V)$,
q_{ktn} : quantity of product n in inventory of node k during period t $\left(\forall k \in K,\ \forall t \in H^0,\ \forall n \in N\right)$,
nv_i^b : maximum number of type b vehicles used by DC i in the planning horizon $(\forall i \in I,\ \forall b \in B)$,
$u_{jtv}^b \in \{0,\ 1,\ \ldots, |J|-1\}$: subtour elimination variable $(\forall j \in J,\ \forall t \in H,\ \forall b \in B,\ \forall v \in V)$,

Model (M1)

$$\begin{aligned} \min \sum_{i\in I}\sum_{e\in E}(OPC_i^e + OPC_i^e \cdot MCQ_i \cdot |H|)\cdot y_i^e + \sum_{i\in I}\sum_{b\in B} FC^b \cdot nv_i^b + \\ \alpha \sum_{i\in I}\sum_{t\in H}\sum_{n\in N} OC_{in}\cdot\ z_{itn} + \beta \sum_{i\in K}\sum_{j\in K\ \mid\ j\neq i}\sum_{t\in H}\sum_{b\in B}\sum_{v\in V} c_{ij}^b\ \cdot x_{ijtv}^b + \\ \theta \sum_{i\in I}\sum_{n\in N}\sum_{t\in H} h_{itn}\ \cdot\ q_{itn} + \gamma \sum_{j\in J}\sum_{n\in N}\sum_{t\in H} h_{jtn}\ \cdot\ q_{jtn} \end{aligned} \tag{1}$$

$$s.t.\ q_{i(t-1)n} + s_{itn} - \sum_{j\in J}\sum_{b\in B}\sum_{v\in V} w^b_{ijtnv} = q_{itn} \quad \forall i \in I, \forall t \in H,\ \forall n \in N \tag{2}$$

$$q_{j(t-1)n} + \sum_{i\in I}\sum_{b\in B}\sum_{v\in V} w^b_{ijtnv} - d_{jtn} = q_{jtn} \quad \forall j \in J,\ \forall t \in H,\ \forall n \in N \tag{3}$$

$$q_{i0n} = IS_{in} \cdot \sum_{e\in E} y^e_i \qquad \forall i \in I,\ \forall n \in N \tag{4}$$

$$q_{j0n} = IS_{jn} \qquad \forall j \in J,\ \forall n \in N \tag{5}$$

$$\sum_{n\in N} (q_{i(t-1)n} + s_{itn}) \le \sum_{e\in E} W^e_i \ \cdot\ y^e_i \quad \forall i \in I,\ \forall t \in H \tag{6}$$

$$\sum_{n\in N} (q_{j(t-1)n} + \sum_{b\in B}\sum_{v\in V} w^b_{ijtnv}) \le\ W_j \qquad \forall i \in I,\ \forall j \in J,\ \forall t \in H \tag{7}$$

$$\sum_{i\in I}\sum_{j\in J}\sum_{n\in N} w^b_{ijtnv} \le\ Q^b \qquad \forall t \in H,\ \forall b \in B,\ \forall v \in V \tag{8}$$

$$s_{itn} \le \max_{e\in E}(W^e_i)\ \cdot\ z_{itn} \qquad \forall i \in I,\ \forall t \in H, \forall n \in N \tag{9}$$

$$\sum_{b\in B}\sum_{v\in V} w^b_{ijtnv} \le f_{ij}\ \cdot\ \sum_{l=t}^{p} d_{jln} \qquad \forall i \in I,\ \forall j \in J,\ \forall t \in H,\ \forall n \in N \tag{10}$$

$$\sum_{n\in N} w^b_{ijtnv} \le \min(Q^b, W_j)\ \cdot\ \sum_{k\in J} x^b_{iktv} \quad \forall i \in I,\ \forall j \in J, \forall t \in H,\ \forall b \in B,\ \forall v \in V \tag{11}$$

$$\sum_{n\in N} w^b_{ijtnv} \le \min(Q^b, W_j)\ \cdot \sum_{k\in J/\{j\}\cup\{i\}} x^b_{kjtv} \quad \forall i \in I,\ \forall j \in J, \forall t \in H,\ \forall b \in B, \forall v \in V \tag{12}$$

$$\sum_{i\in I} f_{ij} = 1 \qquad \forall j \in J \tag{13}$$

$$f_{ij} \le \sum_{e\in E} y^e_i \qquad \forall j \in J,\ \forall i \in I \tag{14}$$

$$\sum_{e\in E} y^e_i \le 1 \qquad \forall i \in I \tag{15}$$

$$\sum_{j\in K\ |\ j\neq i} x^b_{jitv} - \sum_{j\in K\ |\ j\neq i} x^b_{ijtv} = 0 \quad \forall i \in K, \forall t \in H,\ \forall b \in B,\ \forall v \in V \tag{16}$$

$$\sum_{i\in K\ |\ i\neq j}\sum_{b\in B}\sum_{v\in V} x^b_{jitv} \le 1 \quad \forall j \in J,\ \forall t \in H \tag{17}$$

$$\sum_{i\in K\ |\ i\neq j}\sum_{b\in B}\sum_{v\in V} x^b_{ijtv} \le 1 \quad \forall j \in J,\ \forall t \in H \tag{18}$$

$$\sum_{i\in I}\sum_{j\in J} x^b_{ijtv} \le 1 \quad \forall t \in H,\ \forall b \in B,\ \forall v \in V \tag{19}$$

$$x^b_{ijtv} = 0 \qquad \forall i, j \in I,\ \forall t \in H, \forall b \in B, \forall v \in V,\ i \neq j \tag{20}$$

$$\sum_{j\in J}\sum_{v\in V} x^b_{ijtv} \le nv^b_i \qquad \forall i \in I,\ \forall t \in H,\ \forall b \in B \tag{21}$$

$$\sum_{k\in J} x^b_{iktv} + \sum_{k\in K/\{j\}} x^b_{kjtv} -\ f_{ij} \le 1 \quad \forall i \in I,\ \forall j \in J,\ \forall t \in H,\ \forall b \in B,\ \forall v \in V \tag{22}$$

$$\sum_{j\in J} x^b_{ijtv} \leq \sum_{e\in E} y^e_i \qquad \forall i \in I,\ \forall t \in H,\ \forall b \in B,\ \forall v \in V \tag{23}$$

$$\sum_{j\in J}\sum_{t\in H}\sum_{b\in B}\sum_{v\in V} x^b_{ijtv} \geq \sum_{e\in E} y^e_i \qquad \forall i \in I \tag{24}$$

$$u^b_{itv} - u^b_{jtv} + |J| \cdot x^b_{ijtv} \leq |J| - 1 \qquad \forall i, j \in J,\ \forall t \in H, \forall b \in B,\ \forall v \in V,\ i \neq j \tag{25}$$

$$y^e_i \in \{0,1\} \qquad \forall i \in I, \forall e \in E \tag{26}$$

$$f_{ij} \in \{0,1\} \qquad \forall i \in I, \forall j \in J \tag{27}$$

$$s_{itn} \geq 0 \qquad \forall i \in I, \forall t \in H,\ \forall n \in N \tag{28}$$

$$w^b_{ijtnv} \geq 0 \quad \forall i \in I, \forall j \in J,\ \forall t \in H, \forall n \in N, \forall b \in B,\ \forall v \in V \tag{29}$$

$$q_{ktn} \geq 0 \qquad \forall k \in K, \forall t \in H, \forall n \in N \tag{30}$$

$$x^b_{ijtv} \in \{0,\ 1\} \quad \forall i, j \in K, \forall t \in H, \forall b \in B, \forall v \in V, i \neq j \tag{31}$$

$$z_{itn} \in \{0,1\} \quad \forall i \in I, \forall t \in H, \forall n \in N \tag{32}$$

$$nv^b_i \geq 0 \quad \forall i \in I, \forall b \in B \tag{33}$$

$$u^b_{jtv} \geq 0 \qquad \forall j \in J, \forall t \in H, \forall b \in B, \forall v \in V \tag{34}$$

$$\sum_{k\in K\ |\ k\neq j} x^b_{kjtv} \leq \sum_{i\in I}\sum_{n\in N} w^b_{ijtnv} \quad \forall j \in J, \forall t \in H, \forall b \in B, \forall v \in V \tag{35}$$

Regarding objective function, equation (1) is the sum of the opening plus operating cost, using vehicles at least once cost, ordering cost, distribution cost and holding with obsolescence cost.

The set of Eq. (2) and (3) are inventory balance equations for DCs and retailers. The set of Eq. (4) and (5) assigns the initial inventory levels to DCs and retailers. We guarantee the capacity of DCs with constraints (6), the capacity of retailers with constraints (7) and the capacity of vehicles with constraints (8). Constraints (9) ensure the ordering decisions at DCs. Constraints (10) state that retailers can be replenished only by their assigned DCs. Constraints (11) and (12) guarantee the vehicle visits for replenished retailers. Each retailer is assigned to a single opened DC by constraints (13) and (14). Constraints (15) state that each DC can be opened in one of the capacity levels.

Regarding the distribution, constraints (16)-(25) ensure feasible routing. Equations (16)-(18) are traditional vehicle flow conservation constraints. Equations (19) ensure that each vehicle can perform at most one route per period. There cannot be any route between DCs as enforced by equations (20). Constraints (21) find the number of vehicles used by DCs. Constraints (22) state that a retailer can be linked to a DC if it is assigned to that DC. Routes must be related to the opened DCs as enforced by constraints (23) and (24). Constraints (25) are for sub tour elimination. Constraints (26)-(34) state the nature of decision variables.

We guarantee the triangle inequality ($dist_{ij} \leq dist_{ik} + dist_{kj}$) through the parameters used in instance creation, otherwise visits without distribution should be forbidden by adding constraints (35) to the model.

2.2. **Valid inequalities.** M1 model can be strengthened by additional inequalities. However, inequalities for the traveling salesman problem, vehicle routing problem and inventory-routing problem are not valid as stated in Guerrero *et al.* [7]. Nevertheless, we adapted the two valid inequalities proposed for ILRP by Guerrero *et al.* [7] to the MILRP with heterogeneous fleet as follows.

Theorem 2.1. *The inequalities* (36) *are valid for the M1 model.*

$$\min\left(\max_{b\in B}\left(Q^b\right),\max_{e\in E}\left(W_i^e\right)\right)\cdot\sum_{b\in B}\sum_{v\in V}\sum_{j\in J}x_{ijtv}^b\geq\sum_{n\in N}\sum_{j\in J}\sum_{b\in B}\sum_{v\in V}w_{ijtnv}^b\quad\forall i\in I,\ \forall t\in H \tag{36}$$

Proof of Theorem 2.1. If the DC i distributes products to satisfy the demands at period t - i.e., $\sum_{n\in N}\sum_{j\in J}\sum_{b\in B}\sum_{v\in V}w_{ijtnv}^b>0$ - then:

- At least $\sum_{n\in N}\sum_{j\in J}\sum_{b\in B}\sum_{v\in V}w_{ijtnv}^b/\max_{b\in B}\left(Q^b\right)$ vehicles must depart from DC i at period t to satisfy such a demand - i.e., $\sum_{b\in B}\sum_{v\in V}\sum_{j\in J}x_{ijtv}^b\geq\sum_{n\in N}\sum_{j\in J}\sum_{b\in B}\sum_{v\in V}w_{ijtnv}^b/\max_{b\in B}\left(Q^b\right)$, if the capacity of the largest vehicle is tight ($\min\left(\max_{b\in B}\left(Q^b\right),\max_{e\in E}\left(W_i^e\right)\right)=\max_{b\in B}\left(Q^b\right)$).
- At least one vehicle must depart from DC i at period t to satisfy such a demand - i.e., $\sum_{b\in B}\sum_{v\in V}\sum_{j\in J}x_{ijtv}^b\geq\sum_{n\in N}\sum_{j\in J}\sum_{b\in B}\sum_{v\in V}w_{ijtnv}^b/\max_{e\in E}\left(W_i^e\right)$, if the capacity of the largest vehicle is loose ($\min\left(\max_{b\in B}\left(Q^b\right),\max_{e\in E}\left(W_i^e\right)\right)=\max_{e\in E}\left(W_i^e\right)$). □

Theorem 2.2. *The inequalities* (37) *are valid for the M1 model.*

$$\sum_{i\in K\ |\ i\neq j}\sum_{b\in B}\sum_{v\in V}\sum_{l=1}^{t}x_{ijlv}^b\geq\left\lceil\frac{\sum_{n\in N}\sum_{l=1}^{t}d_{jln}-\sum_{n\in N}IS_{jn}}{\max_{b\in B}\left(Q^b\right)}\right\rceil\quad\forall j\in J,\ \forall t\in H \tag{37}$$

Proof of Theorem 2.2. The set of constraints (37) state that the minimal number of times a retailer must be visited up to period t equals to the total demand of that retailer that cannot be satisfied by its initial inventory divided by the capacity of the largest vehicle. □

3. **Sequential heuristic.** We see that only very small-sized instances can be solved exactly through IBM ILOG CPLEX solver within a reasonable computation time after doing preliminary experiments. Therefore, we develop the sequential heuristic method (SHM) to solve all sized MILRP instances within a reasonable time. Table 2 gives a general overview of the SHM. In the SHM, we find the location-allocation and inventory management decisions deterministically through the $findSeqSolWithoutRoutes$ procedure first. Then, we find the routing decisions stochastically through $findAllRoutes$ procedure. Hence, only the routing decisions differ between runs.

In the $findSeqSolWithoutRoutes$ procedure presented in Table 3, we find location-allocation decisions through $findLocAllocDecisions$ procedure first and then, we find the inventory management decisions through the $findInvManDecUsingInvManModel$ procedure. Finally, we check the capacity

levels of opened DCs and set the minimum required capacity level to each opened DC through the *checkAndUpdateOpenedDCsCapLevels* procedure.

TABLE 2. Pseudocode of the sequential heuristic method

Procedure: solveSeq
1: $S_{00} \leftarrow$ findSeqSolWithoutRoutes()
2: **for** *repNum* = 1 to *numOfReps* **do** // for all replications
3: $S_0 = S_{00}$
4: $S^* \leftarrow$ findAllRoutes(S_0)
5: Save the S^* as the best solution found for the replication numbered *repNum*

TABLE 3. Pseudocode of the procedure for finding sequential solution without routes

Procedure: findSeqSolWithoutRoutes
1: $S_{00} \leftarrow$ findLocAllocDecisions()
2: $S_{00} \leftarrow$ findInvManDecUsingInvManModel(S_{00})
3: $S_{00} \leftarrow$ checkAndUpdateOpenedDCsCapLevels(S_{00})
4: return S_{00}

3.1. **Finding the location-allocation decisions.** In the *findLocAllocDecisions* procedure, we first calculate the estimated total costs for each DC to make location decisions. The estimated total cost for a DC is the sum of estimated opening, estimated ordering, and estimated distribution costs for that DC.

For the estimated opening cost for each DC, we take the average of its opening costs. For the estimated ordering cost for a DC named i we do the following calculation. First, we assume that all retailers are assigned to DC i. Then, for each product, we find the number of required orders to satisfy the total demand considering the initial stock of DC i and multiply this number with the ordering cost of product n for this DC. After summing these values, we find the estimated ordering cost for DC i.

The estimated distribution cost for DC i is calculated as follows. For each retailer, we find the number of required distributions times travelling cost to satisfy the retailer demand with minimum capacity vehicle. Then we sum these costs and find the estimated distribution cost for DC i.

After finding the estimated total costs for DCs, DCs are opened by starting from DC with the minimum estimated total cost. While assigning a retailer to a DC, we assume that the retailer's demands are supplied and delivered at the same demand periods, and we check the DC's capacity accordingly. When an opened DC's highest capacity level is exceeded, next DC is opened and filled with the remaining retailers. While assigning the retailers to an opened DC, retailers are sorted in ascending order according to their distances to the opened DC and then assigned in this order. When all the retailers are assigned, the location-allocation process finishes.

3.2. **Finding the inventory management decisions.** We developed the M2 inventory management model to find inventory management decisions exactly. Through the $findInvManDecUsingInvManModel$ procedure, location-allocation decisions are given to the M2 model as input. The M2 model is then created and solved to find the product quantities to be ordered and delivered by opened DCs for all periods. An important assumption made in the M2 model is using only the minimum capacity vehicle for distribution to reduce solution time. This assumption is made with no doubt because, as stated in instance creation parameters, the maximum demand of all retailers can be delivered with the minimum capacity vehicle. M2 model is as follows:

Sets

N : products,
I^o : opened DCs,
J : retailers,
J^i : retailers assigned to opened DC i $(\forall i \in I^o,\ J^i \subseteq J)$,
$K = I^o \cup J$,
$A = \{(i,j); i \in I^o,\ j \in J^i\}$: arcs,
$G = (K, A)$: weighted and directed graph,
$E = \{1,2,3\}$: capacity levels for DCs,
$H = \{1,2,..,\text{p}\}$: periods,
$H^0 = \{0\} \cup H = \{0, 1, 2, .., p\}$: periods including initial period,

Parameters

E_i : capacity level of opened DC i $(\forall i \in I^o, \forall E_i \in E)$,
I_j : DC that retailer j is assigned to $(\forall j \in J, \forall I_j \in I^o)$,
$dist_{ij}$: length of $i-j$ arc in km $(\forall (i,j) \in A)$,
c^0 : unit cost of travelling per km for minimum capacity vehicle,
c^0_{ij} : cost of travelling for $i-j$ arc passed by minimum capacity vehicle $(\forall (i,j) \in A), c^0_{ij} = c^0.\ dist_{ij}$
d_{jtn} : demand of retailer j for product n at period t $(\forall j \in J,\ \forall t \in H,\ \forall n \in N)$,
OC_{in} : ordering cost of product n for opened DC i $(\forall i \in I^o, \forall n \in N)$,
h_{ktn} : total unit holding with obsolescence cost of node k at period t for product n $(\forall k \in K,\ \forall t \in H,\ \forall n \in N)$,
W^E_i : capacity of DC i opened with capacity level E_i $(\forall i \in I^o, \forall E_i \in E)$,
W_j : capacity of retailer j $(\forall j \in J)$,
Q^0 : capacity of minimum capacity vehicle,
FC^0 : cost of using minimum capacity vehicle at least once,
IS_{kn} : initial inventory of node k for product n $(\forall k \in K,\ \forall n \in N)$,
α : weight factor for ordering cost,
β : weight factor for distribution cost,
θ : weight factor for inventory and obsolescence cost total for DCs,
γ : weight factor for inventory and obsolescence cost total for retailers,

Decision variables

$$z_{itn} : \begin{cases} 1 & \text{if product } n \text{ is ordered by opened DC } i \text{ at period t} \\ & (\forall i \in I^o,\ \forall t \in H,\ \forall n \in N) \\ 0 & \text{otherwise} \end{cases}$$

$$\hat{x}_{ijtv} : \begin{cases} 1 & \text{if retailer j is replenished by its assigned DC } I_j \text{ at period } t \\ & \text{at period } t, \left(\forall I_j \in I^o, \forall j \in J^{I_j},\ \forall t \in H\right) \\ 0 & \text{otherwise} \end{cases}$$

s_{itn} : quantity of product n ordered by opened DC i at period t $(\forall i \in I^o,\ \forall t \in H,\ \forall n \in N)$,
w_{I_jjtn} : quantity of product n delivered to retailer j from its assigned DC I_j at period t $\left(\forall I_j \in I^o,\ \forall j \in J^{I_j},\ \forall t \in H,\ \forall n \in N\right)$,
q_{ktn} : quantity of product n in inventory of node k during period t $\left(\forall k \in K,\ \forall t \in H^0,\ \forall n \in N\right)$,
nv_i^0 : maximum number of minimum capacity vehicles used by opened DC i in the planning horizon $(\forall i \in I^o)$,

Model$(M2)$

$$\min \sum_{i\in I^o} FC^0 \cdot nv_i^0 + \alpha \sum_{i\in I^o}\sum_{t\in H}\sum_{n\in N} OC_{in}\cdot\ z_{itn} + \beta \sum_{j\in J^{I_j}}\sum_{t\in H} 2\cdot c^0_{I_jj}\ \cdot \hat{x}_{I_jjt} + \theta \sum_{i\in I^o}\sum_{n\in N}\sum_{t\in H} h_{itn}\ \cdot\ q_{itn} + \gamma \sum_{j\in J}\sum_{n\in N}\sum_{t\in H} h_{jtn}\ \cdot\ q_{jtn} \tag{38}$$

$$s.t.\quad q_{I_j(t-1)n} + s_{I_jtn} - \sum_{j\in J^{I_j}} w_{I_jjtn} = q_{I_jtn} \quad \forall I_j \in I^o, \forall t \in H,\ \forall n \in N \tag{39}$$

$$q_{j(t-1)n} + w_{I_jjtnv} - d_{jtn} = q_{jtn} \qquad \forall I_j \in I^o, \forall j \in J^{I_j},\ \forall t \in H,\ \forall n \in N \tag{40}$$

$$q_{i0n} = IS_{in} \qquad \forall i \in I^o,\ \forall n \in N \tag{41}$$

$$q_{j0n} = IS_{jn} \qquad \forall j \in J,\ \forall n \in N \tag{42}$$

$$\sum_{n\in N} \left(q_{i(t-1)n} + s_{itn}\right) \le W_i^{E_i} \qquad \forall i \in I^o, \forall t \in H \tag{43}$$

$$\sum_{n\in N} \left(q_{j(t-1)n} + w_{I_jjtn}\right) \le\ W_j \qquad \forall I_j \in I^o,\ \forall j \in J^{I_j},\ \forall t \in H \tag{44}$$

$$\sum_{n\in N} w_{I_jjtn} \le\ Q^0 \qquad \forall I_j \in I^o, \forall j \in J^{I_j}, \forall t \in H \tag{45}$$

$$s_{itn} \le W_i^{E_i}\ \cdot\ z_{itn} \qquad \forall i \in I^o,\ \forall t \in H, \forall n \in N \tag{46}$$

$$\sum_{n\in N} w_{I_jjtn} \le \min(Q^0, W_j)\ \cdot\ \hat{x}_{I_jjt} \quad \forall I_j \in I^o,\ \forall j \in J^{I_j}, \forall t \in H \tag{47}$$

$$\sum_{j\in J^{I_j}}\sum_{n\in N} w_{I_jjtn} \le nv^0_{I_j}\ \cdot\ Q^0 \qquad \forall I_j \in I^o,\ \forall t \in H \tag{48}$$

$$z_{itn} \in \{0,1\} \quad \forall i \in I^o, \forall t \in H, \forall n \in N \tag{49}$$

$$\hat{x}_{I_jjt} \in \{0,\ 1\} \quad \forall I_j \in I^o, \forall j \in J^{I_j},\ \forall t \in H \tag{50}$$

$$s_{itn} \geq 0 \qquad \forall i \in I^o, \forall t \in H, \ \forall n \in N \tag{51}$$

$$w_{I_j jtn} \geq 0 \quad \forall I_j \in I^o, \forall j \in J^{I_j}, \ \forall t \in H, \forall n \in N \tag{52}$$

$$q_{ktn} \geq 0 \qquad \forall k \in K, \forall t \in H^0, \forall n \in N \tag{53}$$

$$nv_i^0 \in Z^+ \qquad \forall i \in I^o \tag{54}$$

Regarding the objective function; equation (38) is the sum of using minimum capacity vehicles at least once cost, ordering cost, distribution cost and holding with obsolescence cost. The set of Eq. (39) and (40) are inventory balance equations for opened DCs and retailers. The set of Eq. (41) and (42) assigns the initial inventory levels to the opened DCs and retailers. With constraints (43) capacity of opened DCs, with constraints (44) capacity of retailers and with constraints (45) capacity of vehicles are guaranteed. Constraints (46) ensure the ordering decisions of opened DCs. The relationship between product delivery and the amount of product delivered is ensured by constraints (47). Constraints (48) find the maximum number of minimum capacity vehicles used by opened DCs. Constraints (49)-(54) state the nature of decision variables.

Two valid inequalities adapted to the M1 model are also adapted to the M2 model as follows.

Theorem 3.1. *The inequalities (55) are valid for the M2 inventory management model.*

$$\min\left(Q^0, W_{I_j}^{E_{I_j}}\right) \cdot \sum_{j \in J^{I_j}} \hat{x}_{I_j jt} \geq \sum_{n \in N} \sum_{j \in J^{I_j}} w_{I_j jtn} \qquad \forall I_j \in I^o, \forall t \in H \tag{55}$$

Proof of Theorem 3.1. If an opened DC i distributes products at period t to satisfy the demand of retailers assigned to it - i.e., $\sum_{n \in N} \sum_{j \in J^{I_j}} w_{I_j jtn} > 0$ - then:

- At least $\sum_{n \in N} \sum_{j \in J^{I_j}} w_{I_j jtn} / \ Q^0$ number of deliveries must be made by DC i at period t to satisfy such a demand - i.e., $\sum_{j \in J^{I_j}} \hat{x}_{I_j jt} \geq \sum_{n \in N} \sum_{j \in J^{I_j}} w_{I_j jtn} / \ Q^0$, if the capacity of the minimum capacity vehicle is tight ($\min\left(Q^0, W_{I_j}^{E_{I_j}}\right) = Q^0$).
- At least one delivery must be made by DC i at period t to satisfy such a demand - i.e., $\sum_{j \in J^{I_j}} \hat{x}_{I_j jt} \geq \sum_{n \in N} \sum_{j \in J^{I_j}} w_{I_j jtn} / \ W_{I_j}^{E_{I_j}}$, if the capacity of the minimum capacity vehicle is loose $\left(\min\left(Q^0, W_{I_j}^{E_{I_j}}\right) = W_{I_j}^{E_{I_j}}\right)$. □

Theorem 3.2. *The inequalities (56) are valid for the M2 inventory management model*

$$\sum_{l=1}^{t} \hat{x}_{I_j jl} \geq \left\lceil \frac{\sum_{n \in N} \sum_{l=1}^{t} d_{jln} - \sum_{n \in N} IS_{jn}}{Q^0} \right\rceil \qquad \forall I_j \in I^o, \forall j \in J^{I_j}, \forall t \in H \tag{56}$$

Proof of Theorem 3.2. The set of constraints (56) state that the minimal number of deliveries made to a retailer up to period t equals to the total demand that cannot be satisfied with the initial inventory of that retailer divided by the capacity of minimum capacity vehicle. □

3.3. **Finding vehicle routes.** Vehicle routes of the sequential solution are found using an algorithm based on Pasha *et al.* [15]. The significant differences between our algorithm and the referenced algorithm are explained in the relevant parts. The $findAllRoutes$ procedure presented in Table 4 finds the vehicle routes for all open DCs in the input solution (*sol*).

In the procedure first, for each open DC, we find the routes for all periods independently. Hence, we obtain routes requiring different (same with a small probability) vehicle fleets for each period. Next, for each open DC we find the fixed fleet (fleet that will be used for all periods) using four pre-defined fixed fleet finding strategies, and if the found fixed fleet has never been tried before, we run the following process. The aim of using these strategies is to vary the tried fixed fleets, balance the fleets required for each period and decrease the distribution plus using vehicles at least once cost for each open DC. In this process with the strategy in turn, we find the fixed fleet for open DC i ($newFleetForOpenDC_i$). Next, for each period, we compare $newFleetForOpenDC_i$ with the fleet required by the current routes of that period. If the fleet of open DC i at period t is not a subset of $newFleetForOpenDC_i$, we try to find feasible routes with $newFleetForOpenDC_i$. After this process, we check the feasibility of the routes. If the routes are not feasible for all periods, we use the original feasible routes and the original fleet. Otherwise, we check the distribution plus using vehicles at least once cost of open DC i and if we see a decrease in this cost, we assign the new routes and $newFleetForOpenDC_i$ to the open DC i.

TABLE 4. Pseudocode of the procedure for finding vehicle routes

Procedure: findAllRoutes*(sol)*

1: **for** $i = 1$ to *sol.getNumOfOpenDCs()* **do** // For all open DCs in the solution
2: Assign the next open DC to $openDC_i$
3: **for** $t = 1$ to *numOfPeriods* **do** // For all periods
4: **If** there is a distribution (dist.) to $openDC_i$ at period t
5: Find the routes for $openDC_i$ at period t using *findRoutes* procedure
6: Add the fleet of $openDC_i$ to the new created *triedFleetsForOpenDCI* list

7: **for** *strInd* $= 1$ to *numOfStrategies* **do** //For all of the fleet finding strategies
8: Find a new fleet ($newFleetForOpenDC_i$) for $openDC_i$ with the strategy in turn

9: **If** $newFleetForOpenDC_i$ is not included in the *triedFleetsForOpenDCI*list
10: Add $newFleetForOpenDC_i$ to the *triedFleetsForOpenDCI* list
11: **for** $t = 1$ to *numOfPeriods* **do** // For all periods
12: **If** there is a distribution to $openDC_i$ at period t
13: **If** the fleet of $openDC_i$ at period t is not a subset of $newFleetForOpenDC_i$
14: Find and assign routes to $openDC_i$ for period t with $newFleetForOpenDC_i$

15: **If** all of the new routes of $openDC_i$ is feasible
16: **If** dist. plus fixed cost of new routes for $openDC_i$ is less than the original cost
17: Assign the new routes to $openDC_i$

The $findAllRoutes$ method presented in Table 4 differs from the method proposed by Pasha *et al.* [15] in steps 6-20. The most crucial difference is using four strategies consecutively in this study instead of using just one randomly generated strategy as in the Pasha *et al.* [15]. The first three strategies are Average Plus Large Vehicle Capacity (ALVC), Average Plus Highest Frequency Fraction Vehicle

Capacity (AHFFVC) and Vehicle Mix from the Largest Day (VMLD) strategies that give the first three best results as stated in Pasha *et al.* [15]. ALVC and AHFFVC strategies find the average number of each vehicle type over all periods and add vehicles until the required capacity is reached. The VMLD strategy uses the same fleet as found by the best solution on the period that has the highest delivery amount. The algorithmic details of these strategies are given in the referenced study. We propose a fourth strategy which is the rounded-up form of the ALVC strategy.

We find the routes of an open DC for a specific period t under the constraint of fixed fleet or not with $findRoutes$ procedure presented in Table 5. The procedure is based on the tabu search meta-heuristic proposed by Glover and Laguna [5]. Neighborhood solutions are found through the shift and swap moves. Aspiration criterion in tabu search defines the situations when the tabu criterion can be overridden. In our implementation, we select aspiration criterion as leading a new best solution. Infeasible solutions, in which the amount of product transported on a route is greater than the assigned vehicle's capacity, are also allowed and penalized with the *beta* parameter. $ALFA$ parameter updates the *beta* parameter according to the current solution's feasibility.

With SEH (shrinking and expanding) sub-procedures used in the $findRoutes$ procedure, routes taken as input are shrunken or expanded for diversification purpose. We use the filling degrees of routes to make shrinking or expanding decision. When a route's filling degree is above the pre-determined filling degree threshold (*fillingDegreeThres*), we first assign higher capacity vehicles consecutively until the filling degree is dropped under the $fillingDegreeThres$. After assigning the highest capacity vehicle to the route, if the filling degree does not fall under the $fillingDegreeThres$, we divide the route into new routes. On the other hand, when a route's filling degree is below the $fillingDegreeThres$, we first assign lower capacity vehicles consecutively until its filling degree exceeds the $fillingDegreeThres$. After assigning the lowest capacity vehicle to the route and its filling degree does not exceed the $fillingDegreeThres$, we empty the route by assigning all of its retailers to the other nearest route(s). These operations are made under the fixed fleet constraint in $SEHWithFixedFleet$ procedure and without the fixed fleet constraint in $SEHWithoutFixedFleet$ sub-procedure. SEH sub-procedures used in our study are similar to the SEH procedures used in Pasha *et al.* [15].

The $findRoutes$ procedure differs from the method proposed by Pasha *et al.* [15] with the following major differences. The first difference is our implementation's extra capability that it can also be run under the fixed fleet constraint. The second difference is considering four neighborhood solutions per iteration instead of two to search the neighborhood more effectively. The third difference is the permission of infeasible solutions if the vehicle capacity is not exceeded over a pre-defined ratio. With this capability, we rule over the search space and shorten the computation time. The fourth difference is the continual update of tabu tenure and SEH procedure call frequency with respect to the global iteration counter used in the procedure. With this difference, we try to escape from local minimums, search the neighborhood more effectively and get better solutions.

With $findInitialRoutes$ procedure, we find the initial routing solution of an open DC for a specific period under the fixed fleet or no constraint. If the procedure is called without the fixed fleet constraint, we create a new route and assign a random

TABLE 5. Pseudocode of the procedure that finds the routes for a specific DC at a given period

Procedure: findRoutes ($DC\ openDC_i, short\ t, short[]\ fixedFleet$)

1: $S_0 \leftarrow$ findInitialRoutes($openDC_i, t, fixedFleet$)
2: **If**($isFeasible(S_0)$) //Initialize the best solution if necessary
3: $S^* = S_0$
4: $globalCounter$ = 6000; $sEHCallFreq$ = 30; // Initialize parameters
5: $beta = PENALTYFACTOR$; $sEHCallCounter$ = 0;
6: **while**($globalCounter > 0$) **do** // Main loop of tabu search
7: $globalCounter = globalCounter$ – 1;
8: updateSEHCallFreqAndTabuTenure*(globalCounter);*
9: $moveList \leftarrow$ findMoves(S_0) // Find the available moves
10: $bestMove \leftarrow$ findNonTabuBestMove*(moveList);*
11: $S_0 \leftarrow$ applyMove($S_0, bestMove$);
12: updateTabuList*();* // Update the tabu list
13: **If**($isFeasible(S_0)$ && ($S_0.getCost() < S^*.getCost()$))
14: $S^* = S_0$
15: **If**($sEHCallCounter >= sEHCallFreq$)
16: $sEHCallCounter = 0$
17: **If**($fixedFleet == null$) // Apply the appropriate SEH procedure to S_0
18: $S_0 \leftarrow$ SEHWithoutFixedFleet(S_0, t)
19: **Else**
20: $S_0 \leftarrow$ SEHWithFixedFleet($S_0, t, fixedFleet$)
21: **If**($isFeasible(S_0)$ && ($S_0.getCost() < S^*.getCost()$))
22: $S^* = S_0$
23: $sEHCallCounter = sEHCallCounter + 1$
24: **If**($isFeasible(S_0)$) // Update the *beta* parameter appropriately
25: *beta = beta * (1 - ALFA)*
26: **Else**
27: *beta = beta * (1 + ALFA)*
28: return S^*

vehicle type to this route first. Then we fill this route with retailers that must be replenished at that period. When the vehicle's capacity is filled, we create a new route, assign a random vehicle type to this route and continue the same process until all retailers who should be replenished at that period are assigned to a route. If the procedure is called under the fixed fleet constraint, vehicles assigned to the routes are selected from the fixed fleet and the same process is run. When all the vehicles in the fixed fleet are used and if any retailers without a route assignment exist, they are assigned to the routes, which are sorted in ascending order according to filling degree. So, the feasibility of the routes under fixed fleet constraint is not guaranteed.

4. **Hybrid heuristic.** To find better quality solutions than the sequential heuristic method, decrease the solution time and increase the size of the problems that can be solved in acceptable running times, the hybrid heuristic method (HHM) is developed and used through *solveHybrid* procedure. The pseudocode of the HHM is presented in Table 6.

TABLE 6. Pseudocode of the hybrid heuristic

Procedure: solveHybrid

1: $S_{00} \leftarrow$ findHybridSolWithoutRoutes() // Find the base hybrid solution
2: **for** *repNum* = 1 to *numOfReps* **do** // For all replications
3: Empty the *triedLocAllocConfigsList*
4: $S_0 = S_{00}$ // Assign the base solution to initial solution
5: Add the location allocation decision of S_0 to the *triedLocAllocConfigsList*
6: $S_0 \leftarrow$ findAllRoutes(S_0) // Find all of the routes for initial solution
7: $S^* \leftarrow$ intensify*(*S_0, *triedLocAllocConfigsList)*
8: $S_{new} = S^*$
9: *diverCounter* = 0 // Initialize diversification counter
10: **while** (*diverCounter* < *totalNumOfDiversifications*) **do**
11: $S_{new} \leftarrow$ diversify(S_{new}, *triedLocAllocConfigsList*)
12: $S_{new} \leftarrow$ intensify(S_{new}, *triedLocAllocConfigsList*)
13: **If** (S_{new}*.getCost()* < S^**.getCost())*
14: $S^* = S_{new}$
15: *diverCounter* = *diverCounter* + 1
16: Save S^* as the best solution found for the replication numbered *repNum*

In the *solveHybrid* method, we create the initial solution without routes through the $findHybridSolWithoutRoutes$ procedure and use this solution in all replications. In a replication first, the base solution is assigned to the initial solution. This solution's routes are found through the $findAllRoutes$ procedure, which is the same procedure used in the sequential method. This solution is then improved through the $intensify$ procedure and the best solution found is assigned to the solution S_{new}. Finally, we run the loop, which calls $diversify$ and $intensify$ procedures sequentially for $totalNumOfDiversifications$ times and saves the best solution found for the relevant replication. In the loop, we change the location and allocation decisions of the solution S_{new} through $diversify$ procedure and then improve the new solution through $intensify$ procedure. In order to create different solutions (having different location and allocation decisions), the location and allocation configurations tried before are stored in the $triedLocAllocConfigsList$ which is used and updated by $intensify$ and $diversify$ procedures. In the $diversify$ procedure a new solution is created until we make sure that it is not tried before (not stored in the $triedLocAllocConfigsList$).

4.1. **Creation of the Initial Solution.** As stated previously, we create the initial solution without its routing decisions through the $findHybridSolWithoutRoutes$ procedure. In this procedure first of all, we find the location and allocation decisions of the initial solution by using the same algorithm used in the sequential heuristic. Then, we find the inventory management decisions of the initial solution by letting the quantities of products delivered to the retailers are equal to the corresponding demands of that period if they cannot be supplied from the retailers' initial stocks. Open DCs in the initial solution order products from producers after the consideration of their initial stock levels. Finally, we find the routing decisions of the initial solution through the $findAllRoutes$ procedure. To find the routing decisions fast, we call the $findAllRoutes$ procedure with 0.1 parameter, which is multiplied by *globalCounter* parameter used in this procedure.

4.2. **Intensification.** In the intensification process, the *intensify* procedure presented in Table 7 is used to improve the given solution. This procedure is the hybridization of the simulated annealing and the tabu search. We divide the intensification process into phases by using the initial and freezing temperatures (*INI_TEMP* and *FRE_TEMP)* of the simulated annealing.

At the beginning of the procedure, we find the phase number, which the procedure is currently in. Then, we call the *setIntParams* sub-procedure with this phase number to assign the appropriate values to the probability parameters used in intensification moves. At each temperature, *NUM_OF_ITERS_AT_ALL_TEMPS* repetitions are made. If the current solution is not improved consecutively for *NUM_OF_CONS_TEMPS_SOL_NOT_IMP* times, a random number is generated. If the value of this number is less than or equal to the *ASSIGN_THE_BEST_SOL_RATIO,* the best solution is assigned to the current solution, otherwise ordering decisions of the open DCs are optimized through *updateOrdersWithOrderingModel* sub-procedure.

There are nine intensification moves. The first four of these moves shift deliveries. The fifth move shifts one retailer from its assigned DC to one of the other open DCs. The sixth move swaps two retailers, which are assigned to different open DCs. The seventh move finds the vehicle routes of all open DCs. In the eighth move (*changeDCsCapLevelsIfNecessary*), we change the capacity levels of relevant open DCs accordingly. In the ninth move (*optimizeDCsOrdersIfNecessary*), we optimize the orders of relevant open DCs.

We apply the first seven moves probabilistically. The conditions when the eighth or ninth move is applied as follows. If the intensification phase has been changed and the minimum filling ratio of the open DCs is less than or equal to the *DC_FIL_DEGREE_MIN_LEVEL* or maximum filling ratio of the open DCs is greater than or equal to the *DC_FIL_DEGREE_MAX_LEVEL,* we assign true value to the *changeAllDCSCapLevMove* variable to apply change open DCs' capacity levels move (*changeDCsCapLevelsIfNecessary*) to the relevant open DCs. When the phase number is even, we assign true value to the *optAllDCsOrdersMove* variable to apply optimize open DCs' orders move (*optimizeDCsOrdersIfNecessary*) to the relevant open DCs. Before applying a move its tabu condition is checked and if the move is tabu, it is not applied (no tabu aspiration criterion).

At the end of the intensification procedure, we first find the vehicle routes of the best solution again. Then, we optimize the capacity levels and orders of the open DCs through *optimizeDCsCapLevelsAndOrders* sub-procedure. In the *optimizeDCsCapLevelsAndOrders* sub-procedure, we compare the best solutions found with two different methods and return the better one. In the first method, we optimize the orders of open DCs through *updateOrdersWithOrderingModel* sub-procedure and then minimize the capacity levels of all open DCs. In the second method, we bring down the capacity levels of open DCs one level (if there is any) and then optimize their orders one by one through the *updateOrdersWithOrderingModel* sub-procedure. If the *updateOrdersWithOrderingModel* sub-procedure finds a feasible solution for the DC taken as input, DC's new capacity level remains, otherwise its capacity level is changed to the original one.

4.2.1. *Shift Delivery Move 1.* In this move applied by *applyShiftDeliMove1* sub-procedure, a random retailer j and then a random delivery period t_1 for this retailer is selected. The delivery made at period t_1 is tried to be shifted to the

TABLE 7. Pseudocode of the intensification process

Procedure: intensify

```
1:  S* = S_m                    // Assign the input solution to the best solution
2:  temp = INI_TEMP;    // Initialize the temperature used in the simulated annealing
3:  solNotImpAtConsTempCounter = 0
4:  Initialize the fields used for tabu search
5:  changeAllDCsCapLevMove = false;
6:  optAllDCsOrdersMove = false;    // By default make optAllDCSOrdersMove passive

7:  while (temp > FRE_TEMP) do    // Main loop of the simulated annealing
8:    preIntPhase = curIntPhase
9:    curIntPhase ← findIntPhase(temp)
10:   setIntParams(curIntPhase, S_m.getNumOfOpenDCs)
11:   If (preIntPhase != curIntPhase)    //If the int. phase has been changed
12:     If( S_m.getOpenedDCsMinFilDegree <= DC_FIL_DEGREE_MIN_LEVEL ||
              S_m.getOpenedDCsMaxFilDegree >= DC_FIL_DEGREE_MAX_LEVEL)
13:       changeAllDCsCapLevMove = true
14:     If (curIntPhase % 2 == 0)    // If the intensification phase number is even
15:       optAllDCsOrdersMove = true    // Make optAllDCsOrdersMove active
16:   solImpAtThisTemp = false    // By default make solImpAtThisTemp false
17:   numOfIterAtThisTempCounter = NUM_OF_ITERS_AT_ALL_TEMPS

18:   while (numOfIterAtThisTempCounter > 0) do
19:     Find the move to be applied randomly and apply it to S_m
20:     deltaCost ← calcDeltaCost(S_new,S_m)
21:     If (deltaCost < 0 || Math.exp(-1.0f * deltaCost / temp) > probRegarCost)
22:       S_m = S_new
23:       If (S_m.getCost() < S*.getCost())    // Update the best solution if necessary
24:         S* = S_m
25:         solImpAtThisTemp = true          // Make solImpAtThisTemp true
26:     numOfIterAtThisTempCounter = numOfIterAtThisTempCounter - 1
27:     Update the fields used for tabu search
28:   If (solImpAtThisTemp == true)
29:     solNotImpAtConsTempsCounter = 0
30:   Else
31:     solNotImpAtConsTempsCounter = solNotImpAtConsTempsCounter + 1
32:   If (solNotImpAtConsTempsCounter == NUM_OF_CONS_TEMPS_SOL_NOT_IMP)
33:     solNotImpAtConsTempsCounter = 0
34:     If (getRandomFloat() <= ASSIGN_THE_BEST_SOL_RATIO)
35:       S_m = S*    // Assign the best solution to the current solution
36:     Else
37:       S_m ← updateOrdersWithOrderingModel(S_m, DC_CAP_MULTIPLIER)
38:       If (S_m.getCost() < S*.getCost())    // If the best solution is improved
39:         S* = S_m
40:   temp = temp * COOLING_SPEED        // Update the current temperature
41: findAllRoutes( S* )    // Find all of the routes for the best solution again
42: S* ← optimizeDCsCapLevelsAndOrders(S*)
43: return S*
```

randomly selected new period t_2. We apply this move when the move is not tabu regarding the retailer j, t_1 and t_2 periods. In the move, we find the amounts of products for each product type, which will be shifted to the period t_2 first. While finding these amounts we consider if there is a route to the retailer j at t_2, if the

vehicle type of this route could be changed, if a new route specific to the retailer j on t_2 should be created. After these considerations, we also check the capacity of the DC to which retailer j is assigned (I_j), the capacity of the retailer j and stock out situations. Then, we adjust the amounts of products for each type, which will be shifted to the t_2 period. After these, we also adjust the ordering and delivery plan of the DC I_j. Finally, we make the necessary updates to the routes for t_1 and t_2 periods. According to the pre-determined probabilities, we bring down the capacity level of the DC I_j and find the routes departing from DC I_j again. We see that this move's applicability ratio is around 0.4 and the improvement ratio of the best solution after applying this move is around 0.03.

4.2.2. *Shift Delivery Move 2.* The move applied by *applyShiftDeliMove2* sub-procedure is the two-shift delivery operator proposed by Zhang *et al.* [19]. The representation of this move is given below.

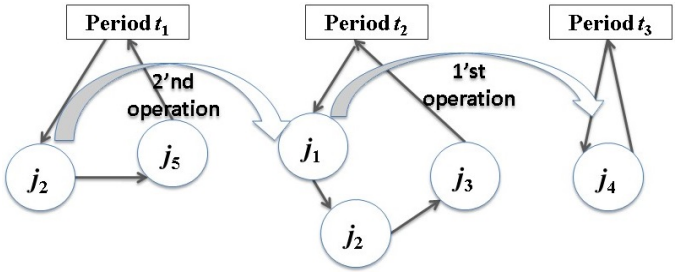


FIGURE 2. Representation of the shift delivery move 2

In this move, we select two random retailers, j_1 and j_2 which share the same route for a random period t_2. We then try to shift the delivery made to the retailer j_1 at period t_2 to the period t_3. After making space in the shared route, we try to shift the delivery made to the retailer j_2 at period t_1 to the period t_2. As we can see, this move is the consecutive application of shift delivery move 1. We apply these moves if the moves are not tabu regarding the retailers and periods. According to the pre-determined probabilities, we bring down the DC's capacity level to which retailers j_1 and j_2 are assigned and find the routes departing from this DC again.

Regarding the results, we see that this move's applicability ratio is around 0.2 and the best solution's improvement ratio after applying this move is around 0.02.

4.2.3. *Shift Delivery Move 3.* The move applied by *applyShiftDeliMove3* sub-procedure is the shift-delivery-deepest operator proposed by Zhang *et al.* [19]. In this move a random retailer j and then a random delivery period t_1 for this retailer is selected. Then, we try to shift the total delivery made at period t_1 to the other delivery periods of this retailer. Before the shift operations, we order the other delivery periods in decreasing the delivery amount. Delivery shift operations are made as in the shift delivery move 1. We apply this move is if the move is not tabu regarding the retailer j and t_1 period. According to the pre-determined probabilities, we bring down the DC's capacity level to which retailer j is assigned and find the routes departing from this DC again.

Regarding the computational results, we see that this move's applicability ratio is around 0.6 and the best solution's improvement ratio after applying this move is around 0.03.

4.2.4. *Shift Delivery Move 4.* The move applied by $applyShiftDeliMove4$ sub-procedure is the consecutive application of the shift delivery move 3 to the retailers that share a randomly selected route. This move aims to empty and delete a selected route. According to the pre-determined probabilities, we bring down the DC's capacity level to which retailers are assigned and find the routes departing from this DC again. Regarding the computational results, we see that this move's applicability ratio is around 0.34 and the best solution's improvement ratio after applying this move is around 0.03.

4.2.5. *Shifting a Retailer Move.* When there are more than one open DCs in the current solution, with the $applyShiftRetaMove$ sub-procedure, we select a retailer j randomly and try to assign this retailer to an open DC I_k, which is different from its assigned DC (I_j). In this move, we cancel the orders and deliveries made by DC I_j to supply retailer j's demands first. Then, the deliveries made by DC I_j to retailer j are tried to be made by DC I_k. We try to make the deliveries in the original amounts and at the original delivery periods during this process. If we could not make the deliveries precisely at the original periods, we try to make the remaining deliveries at other periods. While doing this process, we check stock outs, the capacity of retailer j, and the capacity of DC I_k. After making the necessary adjustments, it is possible to have an infeasible solution. If we get a feasible solution, we create a new location-allocation configuration for the new solution and look for it in the $triedLocAllocConfigsList$. While comparing two configurations, we check if they have the same open DC(s) first. If that is, then we compare their assigned retailers. While comparing the assigned retailers, we try to see that half of the assigned retailers of two compared DCs match. If that is so for all open DC(s), then we say that these two configurations are equal. We add the new configurations to the $triedLocAllocConfigsList$. According to the pre-determined probabilities, we bring down the capacity level of the DC I_j and find the routes departing from DCs I_j and I_k again. Regarding the computational results, we see that this move's applicability ratio is around 0.4 and the best solution's improvement ratio after applying this move is around 0.001.

4.2.6. *Swapping Two Retailers Move.* When there are more than one open DCs in the current solution, with $applySwapRetaMove$ sub-procedure two randomly selected retailers j_1 and j_2, which are assigned to different open DCs I_{j_1} and I_{j_2} are tried to be swapped. This move is the consecutive application of two shift retailer moves, so we apply the same methodology. In this move, after making the necessary adjustments, it is possible to have an infeasible solution. If we get a feasible solution, we create a new location-allocation configuration for the new solution and look for it in the $triedLocAllocConfigsList$. If the list does not contain the new configuration, we add it to the list. On the other hand, if we get an infeasible solution, we cancel this move's application. According to the pre-determined probabilities, we bring down the capacity levels of the open DCs I_{j_1} and I_{j_2} and find the routes of these open DCs again.

Regarding the computational results, we see that this move's applicability ratio is around 0.25 and the best solution's improvement ratio after applying this move is around 0.0003.

4.2.7. *Finding All of the Routes Move.* In this move, which is applied by $findAllRoutes$ sub-procedure, we find the routes of the open DCs again. Regarding

the computational results; we see that the improvement ratio of the best solution after applying this move is around 0.002.

4.2.8. *Changing the Open DCs' Capacity Levels Move.* This move, which is called by $changeDCsCapLevelsIfNecessary$ sub-procedure, is in fact a combination of moves and could only be applied when the intensification phase has changed. In the move, the capacity levels of open DCs whose filling ratio is less than or equal to the *DC_FIL_DEGREE_MIN_LEVEL* are tried to be brought down and the capacity levels of the opened DCs whose filling ratio is greater than or equal to the *DC_FIL_DEGREE_MAX_LEVEL* are tried to be brought up. We change the capacity levels of the correspondent open DCs provided that the move is not tabu. Regarding the computational results, we see that the best solution's improvement ratio after applying this move is around 0.0005.

4.2.9. *Optimizing the Open DCs' Orders Move.* This move, which is called by $optimizeDCsOrdersIfNecessary$ sub-procedure is in fact a combination of moves and could only be applied when the intensification phase number is even. This move aims to optimize the orders and try to get free space in open DCs whose filling ratio is greater than or equal to the *DC_FIL_DEGREE_MAX_LEVEL.* In the $optimizeDCsOrdersIfNecessary$ sub-procedure, optimization of the orders of regarding DCs takes place consecutively, DC by DC, and done with $updateOrdersWithOrderingModel$ sub-procedure. This sub-procedure takes a solution, open DCs list (in this case, only one open DC is stored in this list) and *DC_CAP_MULTIPLIER* parameters as input. After taking these inputs, we create the M3 mathematical model for the open DC in the list. DC's capacity constraint is multiplied by the *DC_CAP_MULTIPLIER* parameter. After solving the model, if we cannot get a feasible solution, the original solution taken as input is returned, otherwise the solution, which contains the open DC with optimized orders is returned. Regarding the computational results, we see that the best solution's improvement ratio after applying this move is around 0.09. M3 model is as follows:

Sets

N : products,
I^o : open DCs,
$E = \{1,2,3\}$: capacity levels for DCs,
$H = \{1,2,..,p\}$: periods,
$H^0 = \{0\} \cup H = \{0, 1, 2, .., p\}$: periods including initial period,

Parameters

D_{itn} : quantity of product n delivered from open DC i at period t $(\forall i \in I^o,\ \forall t \in H,\ \forall n \in N)$,
E_i : capacity level of open DC i $(\forall i \in I^o, E_i \in E)$,
OC_{in} : ordering cost of product n for DC i $(\forall i \in I^o, \forall n \in N)$,
h_{itn} : total unit holding with obsolescence cost of node i at period t for product $n\,(\forall i \in I^o,\ \forall t \in H,\ \forall n \in N)$,
$W_i^{E_i}$: capacity of DC i opened with capacity level E_i $(\forall i \in I^o, \forall E_i \in E)$,
IS_{in} : initial inventory of node i for product $n\,(\forall i \in I^o,\ \forall n \in N)$,
α : weight factor for ordering cost,

θ : weight factor for inventory and obsolescence cost total for DCs,

Decision variables

$$z_{itn} : \begin{cases} 1 & \text{if product } n \text{ is ordered by open DC } i \text{ at period t} \\ & (\forall i \in I^o,\ \forall t \in H,\ \forall n \in N) \\ 0 & \text{otherwise} \end{cases}$$

s_{itn} : quantity of product n ordered by open DC i at period t $(\forall i \in I^o,\ \forall t \in H,\ \forall n \in N)$,

q_{itn} : quantity of product n in inventory of open DC i during period t $\left(\forall i \in I^o,\ \forall t \in H^0,\ \forall n \in N\right)$,

Model$(M3)$

$$\min \alpha \sum_{i \in I^o} \sum_{t \in H} \sum_{n \in N} OC_{in} \cdot z_{itn} + \theta \sum_{i \in I^o} \sum_{n \in N} \sum_{t \in H} h_{itn} \cdot q_{itn} \tag{57}$$

$$s.t. \quad q_{i(t-1)n} + s_{itn} - q_{itn} = D_{itn} \quad \forall i \in I^o, \forall t \in H,\ \forall n \in N \tag{58}$$

$$q_{i0n} = IS_{in} \quad \forall i \in I^o,\ \forall n \in N \tag{59}$$

$$\sum_{n \in N} (q_{i(t-1)n} + s_{itn}) \leq DC_CAP_MULTIPLIER \cdot W_i^{E_i} \quad \forall i \in I^o,\ \forall t \in H \tag{60}$$

$$s_{itn} \leq DC_CAP_MULTIPLIER \cdot W_i^{E_i} \cdot z_{itn} \quad \forall i \in I^o,\ \forall t \in H, \forall n \in N \tag{61}$$

$$z_{itn} \in \{0,1\} \quad \forall i \in I^o, \forall t \in H, \forall n \in N \tag{62}$$

$$s_{itn} \geq 0 \quad \forall i \in I^o, \forall t \in H,\ \forall n \in N \tag{63}$$

$$q_{itn} \geq 0 \quad \forall i \in I^o, \forall t \in H,\ \forall n \in N \tag{64}$$

Regarding the objective function, equation (57) is the sum of ordering cost and holding with obsolescence cost. The set of Eq. (58) are inventory balance equations for open DCs. The set of Eq. (59) assigns the initial inventory levels to open DCs. With constraints (60) we guarantee the capacity of open DCs. Constraints (61) ensure the ordering decisions of open DCs. Constraints (62)-(64) state the nature of decision variables.

4.3. **Diversification.** In the diversification ($diversify$) procedure, we change the input solution's location-allocation decision until we get a new solution with different location-allocation decision. We check the $triedLocAllocConfigsList$ to get a solution with different location-allocation configuration. When the new solution is created, this solution's location-allocation configuration is added to the $triedLocAllocConfigsList$.

To create new solutions, we apply one of the two moves to the input solution after comparing the randomly generated float number and the value of *CLOSE_OPEN_DC_RATIO* parameter. In the first move which is applied when the randomly generated float number is less than or equal to the value of *CLOSE_OPEN_DC_RATIO* parameter, we try to close one of the open DCs (I_c). While doing this operation we try to assign the retailers of this DC to

the closed DC(s) that will be opened or to the other open DC(s). To make this decision, we compare a randomly generated float number and the value of the *CLOSED_DCS_PRI_RATIO* parameter. If the generated float number is less than or equal to the value of *CLOSED_DCS_PRI_RATIO* parameter, we try to assign the retailers to the new opened DC(s); otherwise, we try to assign them to the already opened DC(s). In the second move which is applied when the randomly generated float number is greater than the value of the *CLOSE_OPEN_DC_RATIO* parameter, we select one of the opened DC(s) (I_c) and try to assign a fraction of its retailers to the other already opened DC(s) or close DC(s) which will be opened. The number of retailers that will be randomly selected from DC I_c is found by multiplying the number of total assigned retailers and the *ASSIGN_RETA_RATIO* parameter. In these moves, before assigning the retailers to the already opened DCs or new opened DCs, we maximize the capacity levels of the DCs to make the retailer assignments effectively.

In the diversification moves, we select the open DC I_c randomly using one of the two roulette-wheel selection methods. In the first method, we use the capacity usage information of open DCs (lower capacity usage means higher probability to be chosen), whereas in the second method, we use the estimated total cost of the open DCs found by the $findLocAllocDecisions$ procedure (higher estimated total cost means higher probability to be chosen). While choosing the DC(s) in both moves to be opened, we randomly generate a float number and compare it with the value of *CLOSED_DCS_RAND_OPENING_RATIO* parameter. If the randomly generated float number is less than or equal to the value of the *CLOSED_DCS_RAND_OPENING_RATIO* parameter, the DC(s) that will be opened are selected randomly; otherwise they are ordered by ascending according to their estimated total cost and then selected consecutively.

At the end of the diversification procedure; we minimize the capacity levels of all open DCs, optimize the ordering decisions of the open DCs whose filling ratio are greater than or equal to the *DC_FIL_DEGREE_MAX_LEVEL* consecutively through the $updateOrdersWithOrderingModel$ sub-procedure and refind the routes of the open DCs to which new retailers are assigned or from which some retailers are shifted.

5. Computational study.

We coded the mathematical model and heuristic methods in Java programming language and solved the mathematical models through IBM ILOG CPLEX 12.8.0 solver. We keep PRESOLVE and HEURISTIC options of the solver open. Except for the results in Table 19, we took all runs on a PC with Intel Xeon E5-2640 v4 (32 2.40 GHz cores) processor, 64 GB RAM and 64 bit Windows 10 Professional operating system.

5.1. Instances.

Since there are no available benchmark instances for the MILRP, we generated random instances whose parameters are based on real life information and the Guerrero *et al.* [7] study. We name the instances as $I - J - H - N - ID$, where I is the number of potential DCs, J is the number of retailers, H is the number of periods, N is the number of products and ID is the identification number for the instance.

We generate the demand of retailer j for product n at period t through Normal distribution: $d_{jtn} \sim \text{Norm}(\mu_{jn}, \sigma_{jn})$, where $\mu_{jn} \in [5, 15]$ and $\sigma_{jn} \in [0, 5]$. We locate all the nodes (Producers, DCs and retailers) randomly in an area of size 100 x 100. The length of the $i - j$ arc in km, $dist_{ij}$, is

$\text{NINT}\left(10\ .\ \sqrt{(X_i - X_j)^2 + (Y_i - Y_j)^2}\right)$. NINT (.) function approximates its parameter to the closest integer value. We generate the unit inventory holding cost of product n for a single period $t \in H$ randomly in the interval [0.1, 0.4] for DCs $i \in I$ and in the interval [0.5, 0.9] for retailers $j \in J$. Besides; we generate unit obsolescence cost ε randomly using interval [0.01, 0.02]. Total unit holding plus obsolescence cost of node $k \in K$ in period $t \in H$ for product $n \in N$, h_{ktn}, is $h_{kt(t+1)n} + \epsilon$.

The capacity of DC i with capacity level e, W_i^e, is $\text{NINT}\left(DCFWQ^e\ .\ DCVWQ\ .\ totD\ /\ |I|\right)$ where $DCFWQ^e$ is constant capacity quotient for capacity level $e, DCVWQ$ is dynamic capacity quotient and $totD$ (total demand) is $\sum_{j\in J}\sum_{t\in H}\sum_{n\in N} d_{jtn}$. Constant capacity quotients for DC capacity levels are {1, 1.5, 2} respectively. We generate $DCVWQ$ randomly in the interval [0.6, 1.2]. The opening cost of DC i with capacity level e, OPC_i^e, is $\text{NINT}\left(DCVOC\ .\ OCFQ^e\right)$. $DCVOC$ is dynamic opening cost and is generated randomly in the interval [10000, 16000]. $OCFQ^e$ is constant opening cost quotient for capacity level e and it gets {1, 1.35, 1.65} values for DC capacity levels respectively. We generate the operating cost of DC i per period MCQ_i randomly in the interval [0.01, 0.02]. The capacity of retailer i, Wj, is $\text{NINT}\left(RVQ_j\ .\ \max_{t\in H}\left(\sum_{n\in N} d_{jtn}\right)\right)$ where RVQ_j is a dynamic capacity quotient and generated randomly in the interval [1.5, 3]. We choose the initial inventory of DC i (IS_{in}) randomly from the set $\left\{0,\ \text{NINT}\left(\left(\sum_{j\in J}\sum_{t\in H} d_{jtn}\right)\ /\ |J|\right)\right\}$ and the initial inventory of retailer$j$ (IS_{jn}) randomly from the set $\{0,\ \ d_{j1n}\}$.

The number of vehicles for each vehicle type $|V|$ is equal to the number of retailers $|J|$. Capacity of type b vehicle, Q^b, is $\text{NINT}\left(Q^0\ /\ VFQQ^0\ .\ VFQQ^b\right)$ where the capacity of the minimum capacity vehicle (Q^0) is $\text{NINT}\left(\text{ODP}\left(\mu_{jn}\right).|n|\ .\ VFQQ^0\ .\ VVQQ\right)$. We guarantee that Q^0 is greater than or equal to the maximum retailer demand. ODP (.) function takes a distribution as input and returns the average value of low and high values of that distribution. Constant capacity quotients for vehicle types $VFQQ$ are {1.2, 1.8, 2.4} respectively. We generate dynamic capacity quotient ($VVQQ$) randomly in the interval [1, 2]. Unit costs of travelling per km for vehicle types (c) are {1, 1.2, 1.4} respectively. The cost of travel for the $i-j$ arc passed by type b vehicle is $\text{NINT}\left(c^b.\ dist_{ij}\right)$. The cost of using a type b vehicle at least once (FC^b) is $\text{NINT}\left(VFCQ^b\ .\ VFBUC\right)$. Constant cost quotients for vehicle types ($VFCQ$) are {1.5, 2, 2.5} respectively. We generate constant cost of using type b vehicle at least once ($VFBUC$) randomly in the interval [2000, 4000]. Ordering cost of product n by DC i, OC_{in}, is equal to the sum of DC ordering cost of that product plus length of the arc between product n's producer and DC i. We generate DC i's ordering cost of product n randomly in the interval [50, 100].

5.2. **Preliminary tests.** According to the preliminary tests in which solution time and gap (gap between relaxed model's cost and optimal cost) analysis are made, we see that the proposed two valid inequalities for the M1 model and only the second proposed valid inequality for the M2 model are effective. Therefore, we added these valid inequalities to the relevant models.

Regarding the $PENALTYFACTOR, TABUTENURE$, and $ALFA$ constants used in the vehicle routing algorithm, we take the same low and high values used

in the Pasha *et al.* [15] study to generate and assign the random values to these constants. On the other hand, to find the appropriate values for the constants that substantially affect the performance and solutions' quality, we use "Nearly Orthogonal Linear HyperCubes Design" proposed by Cioppa and Lucas [3]. It is stated in this study that pairwise correlations between factors become nearly zero by using their robust design. we entered the low and high values for each factor into the given excel sheet and generated 16 different parameter sets to use their design. We run the routing algorithm 15 times for each parameter set and assessed all the parameter sets by considering their routing costs' (best, worst, average, and standard deviation). As a result, we find the appropriate values presented in Table 8.

TABLE 8. Values assigned to the important constants used in the vehicle routing algorithm

Constant (Factor)	**Value**
Filling degree threshold (*fillingDegreeThres*) [a]	0.8
Call frequency for SEH functions (*sEHCallFreq*) [b]	30
Number of iterations in the main loop (*globalCounter*) [b]	6000
Extra percentage by which the vehicle capacity can be exceeded (*extraLoadPerc*) [b]	0.6

[a] *fillingDegreeThres* is used in *SEHWithFixedFleet* and *SEHWithoutFixedFleet* procedures.
[b] *sEHCallFreq, globalCounter* and *extraLoadPerc* are used in *findRoutes* procedure.

We use the same methodology described in the previous paragraph to find the appropriate values for the constants *INI_TEMP, FRE_TEMP*, *COOLING_SPEED*, *NUM_OF_ITERS_AT_ALL_TEMPS, NUM_OF_CONS_TEMPS_SOL_NOT_IMP* and *NUM_OF_PHASES*. These constants are used in the simulated annealing part of the intensification process. We take the same low and high values used in Zhang *et al.* [19] for the first five parameters, whereas take 5-20 values for the *NUM_OF_PHASES* parameter. After making the relevant analysis, we find the appropriate values shown in Table 9.

TABLE 9. Values assigned to the important constants, which are used in the intensification process

Constant (Factor)	**Value**
INI_TEMP	45
FRE_TEMP	1
COOLING_SPEED	0.93
NUM_OF_ITERS_AT_ALL_TEMPS	400
NUM_OF_CONS_TEMPS_SOL_NOT_IMP	3
NUM_OF_PHASES	12
ASSIGN_THE_BEST_SOL_RATIO	0.75
DC_FIL_DEGREE_MAX_LEVEL	0.85
DC_FIL_DEGREE_MIN_LEVEL	0.70
DC_CAP_MULTIPLIER	0.80-0.85-0.90 [a]

[a] These values are used at the first half, at the second half and at the final phase, respectively.

After solving many instances and assessing the results, we come up with the appropriate probabilities for the application of intensification moves shown in Table

10 and the appropriate values to be assigned to the important constants (presented in Table 11) used in the diversification process. We assign the rounded-up value of $|I|$ / 5 + $|J|$ / 20 to the *totalNumOfDiversifications* parameter.

TABLE 10. Probabilities for the application of intensification moves

Move's Name	**Application Probabilities**	
	Number of Open DCs > 1	Number of Open DCs = 1
Shift Delivery Move 1	0.30	0.32
Shift Delivery Move 2	0.10	0.11
Shift Delivery Move 3	0.40	0.45
Shift Delivery Move 4	0.10	0.10
Shifting a Retailer Move	0.05	-
Swapping Two Retailers Move	0.03	-
Finding All of the Routes Move	0.02	0.02

TABLE 11. Values assigned to the important constants used in the diversification process

Constant (Factor)	**Value**
CLOSE_OPEN_DC_RATIO	0.8
CLOSED_DCS_PRI_RATIO	0.7-0.3 [a]
ASSIGN_RETA_RATIO	0.6-0.8 [b]
CLOSED_DCS_RAND_OPENING_RATIO	0.7-0.8 [a]

[a] The value assigned to the relevant parameter changes with the phase currently in.
[b] The value assigned to the relevant parameter is generated randomly in this interval.

5.3. **Results.** In Section 2.1, we mentioned that our model for MILRP with heterogeneous fleet (M1 model) is based on the ILRP model proposed by Guerrero *et al.* [7]. We also pointed out that we made some modifications to the based model to decrease the number of decision variables and make the model easily understandable. To compare our model (M1 model) with a new MILRP model, which is totally based on the Guerrero *et al.* [7] ($M1^G$ model), five 3-5-3-2 MILRP instances are solved optimally through ILOG CPLEX solver. The results are given in Table 12. The first column gives the name of the instance. Columns 2-3 show the computation time in seconds elapsed until the optimal solution is found with the $M1^G$ and M1 models.

We see that all the computation times obtained with the $M1^G$ model are higher than the computation times obtained with the M1 model and average computation time for the $M1^G$ model is nearly two times the average computation time for M1 model. These findings prove the effectiveness of our modifications to the $M1^G$ model.

After doing preliminary experiments, we see that only the very small-sized instances could be solved through ILOG CPLEX solver with a time limit of 2 hours. In comparing the solver and the heuristic methods, 20 MILRP instances are solved. We create the instances in a way that we can see both optimal solutions, easy versus difficult instances and different instances concerning the number of potential DCs, retailers, periods and products. The results are given in Table 13.

TABLE 12. Comparison of MILRP models

	$M1^G$ model	M1 model
Instance	CPU (s)	CPU (s)
3-5-3-2-P1	298.6	**170.6**
3-5-3-2-P2	1284.2	**453.7**
3-5-3-2-P3	484.4	**161.7**
3-5-3-2-P4	739.5	**559.9**
3-5-3-2-P5	546.2	**423.9**
Average	670.6	**353.9**

TABLE 13. Computational results for very small-sized MILRP instances

	ILOG				SHM			HHM			
Instance	GAP (1200s)	GAP (7200s)	Cost	CPU (s)	Cost	GAP ILOG	CPU (s)	Cost	GAP ILOG	GAP SHM	CPU (s)
3-5-3-2-P1	0.0	0.0	**35267.9**	220	36952.5	4.6	0.1	36643.6	3.8	-0.8	4.2
3-5-3-2-P2	0.0	0.0	**30858.4**	45	35045.0	11.9	0.1	33006.8	6.5	-6.2	3.0
3-5-3-3-P1	7.9	2.4	**32273.5**	3470	35081.8	8.0	0.1	33085.3	2.5	-6.0	4.1
3-5-3-3-P2	3.3	0.0	**37305.3**	800	37653.9	0.9	0.1	38002.1	1.8	0.9	2.8
3-5-3-3-P3	0.0	0.0	**29107.2**	1379	**29107.2**	0.0	0.2	30585.0	4.8	4.8	3.2
3-5-4-2-P1	12.2	0.0	**31552.6**	3750	31786.9	0.7	0.2	32877.2	4.0	3.3	4.5
3-5-4-2-P2	6.8	2.0	**34855.1**	350	38254.0	8.9	0.2	38554.6	9.6	0.8	4.5
3-5-4-2-P3	10.1	3.3	**33480.8**	4700	36148.7	7.4	0.1	36416.3	8.1	0.7	2.6
3-6-3-2-P1	14.6	9.7	**38556.4**	1966	47082.1	18.1	0.1	43580.0	11.5	-8.0	2.5
3-6-4-3-P1	28.9	14.6	**35581.4**	7200	41216.3	13.7	0.2	38755.4	8.2	-6.3	3.6
5-10-4-2-P1	76.0	56.9	84643.5	7051	66101.1	-28.1	0.3	**65867.8**	-28.5	-0.4	10.9
5-10-5-3-P1	73.0	60.4	114260.7	7185	91432.5	-25.0	0.9	**85159.8**	-34.2	-7.4	18.6
5-12-4-4-P1	78.2	69.4	139721.1	7195	78330.1	-78.4	0.5	**74669.7**	-87.1	-4.9	17.9
5-12-5-3-P1	79.5	72.1	143038.5	7200	76196.6	-87.7	1.1	**74408.4**	-92.2	-2.4	19.1
5-15-3-3-P1	79.0	74.7	154831.3	7190	87291.8	-77.4	0.2	**83214.9**	-86.1	-4.9	12.8
5-15-4-2-P1	80.1	76.9	156552.6	5050	71446.8	-119.1	0.4	**71245.0**	-119.7	-0.3	17.0
5-15-4-4-P1	89.0	74.7	174019.7	5828	**90018.0**	-93.3	0.9	90261.9	-92.8	0.3	18.1
5-15-4-7-P1	86.5	78.0	251935.1	3575	**113323.5**	-122.3	2.1	113618.3	-121.7	0.3	22.1
5-15-5-3-P1	84.6	84.0	243900.1	2450	77672.0	-214.0	1.2	**77267.7**	-215.7	-0.5	22.4
5-20-6-4-P1	100	89.9	436645.1	4805	**119599.3**	-265.1	47.4	120897.7	-261.2	1.1	62.0
Average	45.5	38.4	111919.3	4070.5	61987.0	-51.8	2.8	**60905.9**	-53.9	-1.8	12.8

The first column gives the name of the instance. Columns 2-3 show the gap between lower and upper bound when instances are solved through the solver preset within 1200 s and 7200 s (2 hours). In column four, we see the cost of the best solution found within 7200 s. Column five gives the computation time in seconds elapsed until the best solution is encountered. Columns 6-8 present the average results for SHM in five runs. In these columns, we see the average cost of the solution, the average gap between the cost found by SHM and the cost found by the solver, and the average computation time in seconds. Columns 9-12 present the average results for HHM in five runs. In these columns, we see the average cost of the solution, the average gap between the cost found by HHM and cost found by the solver, the average gap between the cost found by HHM and the cost found by SHM and the average computation time in seconds.

Looking at the values in Table 13, we notice that only 5 out of 20 very small-sized instances could be solved exactly through the solver within a time limit of 7200 s. The quality of the solutions found with the solver gets worse quickly as the problem's size increases. This result proves the difficulty of the MILRP with heterogeneous fleet.

When we look at the results of the first ten instances, which are solved exactly or almost exactly, we see that average gaps between the cost found by the solver and costs found by the SHM and HHM are 7.4% and 6.1%, respectively. We can say that these gaps are acceptable. Besides, on average, the solutions found with SHM are 51.8%, and solutions found with HHM are 53.9% better than the solutions found with the solver regarding all instances. On average HHM outperforms SHM, but SHM outperforms the HHM prominently in two instances. We know that the reason behind this result is the exact inventory management decisions in SHM.

The average computing time for SHM is only 2.8 s, nearly one fifth of the average computing time for HHM (12.8 s). After making an in-depth analysis to understand the reason behind this significant difference, we noticed that SHM uses 99% CPU power of the PC (all (32) cores of the CPU) while solving the inventory management model through a multi-threaded CPLEX solver, which constitutes nearly whole part of the SHM computing time. On the other hand, HHM only uses 5% CPU power of the PC (only one core of the CPU). Nevertheless, computing times for both methods are acceptable.

The analysis regarding the solution time and solution quality of heuristic methods when solving the randomly generated small, medium and large-sized instances is shown in Table 14.

TABLE 14. Computational results for small, medium and large-sized MILRP instances

	SHM				HHM		
Instance	Cost	CPU (s)	IM CPU (s)	PIM (%)	Cost	GAP SHM	CPU (s)
5-50-3-3-P1	172939.9	0.5	0.3	66.67	**167457.3**	-3.3	66.1
5-50-3-3-P2	208332.0	0.5	0.4	76.60	**207016.9**	-0.6	67.7
5-50-3-3-P3	227655.0	0.3	0.2	69.70	**221462.7**	-2.8	65.9
5-50-3-3-P4	230679.9	0.4	0.3	75.68	**215962.2**	-6.8	66.3
5-50-3-3-P5	176528.9	0.4	0.3	78.57	**162825.3**	-8.4	65.1
Average	203227.1	0.4	0.3	73.4	**194944.9**	-4.4	66.2
15-100-5-5-P1	514177.6	141.8	141.5	99.77	**493879.2**	-4.1	414.5
15-100-5-5-P2	502955.3	244.1	243.7	99.84	**485170.5**	-3.7	393.7
15-100-5-5-P3	647847.3	121.8	121.5	99.70	**627396.9**	-3.3	422.6
15-100-5-5-P4	629868.5	193.1	192.8	99.82	**598746.6**	-5.2	433.2
15-100-5-5-P5	578960.6	111.8	111.4	99.69	**560799.6**	-3.2	428.4
Average	574761.9	162.5	162.2	99.8	**553198.6**	-3.9	418.5
20-150-7-7-P1	1214217.0	3402.7	3382.1	99.39	**1129150.2**	-7.5	1182.6
20-150-7-7-P2	1042158.5	3602.7	3602.1	99.98	**983087.1**	-6.0	1294.7
20-150-7-7-P3	1064822.3	3502.9	3475.3	99.21	**1023313.9**	-4.1	1253.8
20-150-7-7-P4	1115968.3	3627.2	3626.6	99.98	**1104168.2**	-1.1	1180.7
20-150-7-7-P5	1041073.8	3727.4	3696.8	99.18	**1004549.8**	-3.6	1182.8
Average	1095648.0	3572.6	3556.6	99.5	**1048853.8**	-4.5	1218.9

The first column gives the name of the instance. Columns 2-5 present the average results of SHM run in five times. In these columns, we see the average cost, average computation time required to solve the instance, the average computation time needed to find inventory management decisions of the instance (IM CPU) and the average percentage of this time (IM CPU) in total computation time. Columns 6-8 present the average results for HHM in five runs. In these columns, we see the

average cost of the solution, the average gap between the cost found by HHM and cost found by SHM and the average computation time in seconds.

When we look at the results for small-sized instances, we see that average computation time for SHM is 0.4 s and 73.4% (0.3 s) amount of this time is used to find the inventory management decisions. On the other hand, the proposed HHM spends 66.2 s for these instances and outperforms SHM by 4.4% on average. Considering the medium-sized instances, we notice that the HHM spends 1.5 times more computation time than SHM, but at the expense of this time, it outperforms SHM by 3.9% on average.

Finally, when we look at the results for large-sized instances, we see that the average computation time for SHM is 3572.6 s (nearly one hour) and the percentage of this time spent on inventory management decisions is similar to the medium-sized instances. On the other hand, HHM spends 1218.9 s (nearly 20 minutes) and outperforms SHM by 4.5% on average. Consequently, we can also say that these CPU times are acceptable for both methods considering the studied problem's difficulty.

After looking at the GAP values in Table 14, we notice that the most significant average gap between the compared solutions' costs occurs at 5-50-3-3-P5 instance. When we seek the reason behind this considerable difference and see that the best solutions found by SHM runs includes two open DCs, whereas the best solutions found by HHM runs includes only one open DC.

The analyses regarding the cost weight factors α, β, θ and γ used in the M1 model's objective function are presented in the following tables. We made all the analyses on medium-sized instances, which are solved five times through the HHM. Tables present the relevant values of the best solutions found.

TABLE 15. Analysis of ordering cost weight factor

	$\alpha = 1, \beta = 1, \theta = 1, \gamma = 1$			$\alpha = 10, \beta = 1, \theta = 1, \gamma = 1$		
Instance	TotCap[a]	TotNumOf Orders[a]	TotInvOf DCs[b]	TotCap[a]	TotNumOf Orders[a]	TotInvOf DCs[b]
15-10-5-5-P1	**5159**	49	**643**	5371	**43**	1199
15-10-5-5-P2	**5277**	46	**834**	6087	**39**	3303
15-10-5-5-P3	**5410**	48	**503**	6431	**38**	2715
15-10-5-5-P4	**5137**	50	**0**	5570	**45**	2148
15-10-5-5-P5	**5392**	48	**644**	5958	**41**	2894

[a]TotCap: Total capacity provided by open DCs, [a]TotNumOfOrders: Total number of orders.
[b]TotInvOfDCs: Total amount of products in inventories of open DCs comprising all periods.

When we look at the results in Table 15, we see that when the value of the ordering cost weight factor changes to the ten times of its original value, the total number of orders made by open DCs decreases directly as expected. Besides, to make fewer orders in high amounts, total capacity provided by open DCs and total amount of products in open DCs' inventories increases.

Regarding the results in Table 16, we see that when the value of the distribution cost weight factor changes to the 10 times of its original value, total number of distributions made by open DCs decreases as expected. Besides, to lower the total distribution cost, the number of open DCs and total capacity provided by open DCs increases. Due to the decrease in the number of distributions, the total amount of products in inventories of open DCs and retailers increases.

When we look at the results in Table 17, we see that when the value of the inventory cost weight factor for DCs changes to 100 times its original value, the total amount of products in open DCs' inventories decreases as expected. Besides, total number of orders made by open DCs increases to decrease the total amount of products in open DCs' inventories.

TABLE 16. Analysis of distribution cost weight factor

	$\alpha=1,\beta=1,\theta=1,\gamma=1$					$\alpha=1,\beta=10,\theta=1,\gamma=1$				
Instance	Num OfOp DCs[a]	Tot Cap[a]	TotNum OfDists[b]	TotInv OfDCs[c]	TotInv OfRets[d]	Num OfOp DCs[a]	Tot Cap[a]	TotNum OfDists[b]	TotInv OfDCs[c]	TotInv OfRets[d]
15-10-5-5-P1	**2**	**5159**	426	**643**	**6404**	3	7226	**322**	6112	13862
15-10-5-5-P2	**2**	**5277**	415	**834**	**7056**	4	7847	**326**	6419	14232
15-10-5-5-P3	**2**	**5410**	380	**503**	**8410**	3	7701	**303**	6576	15643
15-10-5-5-P4	**2**	**5137**	378	**0**	**9575**	3	6701	**299**	3780	14375
15-10-5-5-P5	**2**	**5392**	390	**644**	**8052**	4	9031	**321**	9346	13532

[a]NumOfOpDCs: Number of open DCs, [a]TotCap: Total capacity provided by open DCs.
[b]TotNumOfDists: Total number of distributions.
[c]TotInvOfDCs: Total amount of products in inventories of open DCs comprising all periods.
[d]TotInvOfRets: Total amount of products in inventories of retailers comprising all periods.

TABLE 17. Analysis of inventory cost weight factor for DCs

	$\alpha=1,\beta=1,\theta=1,\gamma=1$		$\alpha=1,\beta=1,\theta=100,\gamma=1$	
Instance	TotNumOfOrders[a]	TotInvOfDCs[b]	TotNumOfOrders[a]	TotInvOfDCs[b]
15-10-5-5-P1	**49**	643	50	**0**
15-10-5-5-P2	**46**	834	50	**0**
15-10-5-5-P3	**48**	503	50	**0**
15-10-5-5-P4	**48**	644	50	**0**
15-10-5-5-P5	**40**	2315	50	**0**

[a]TotNumOfOrders: Total number of orders.
[b]TotInvOfDCs: Total amount of products in inventories of open DCs comprising all periods.

TABLE 18. Analysis of inventory cost weight factor for retailers

	$\alpha=1,\beta=1,\theta=1,\gamma=1$				$\alpha=1,\beta=1,\theta=1,\gamma=100$			
Instance	TotNumOf Orders[a]	TotNum OfDists[a]	TotInvOfDCs[b]	TotInv OfRets[c]	TotNumOf Orders[a]	TotNum OfDists[a]	TotInvOfDCs[b]	TotInv OfRets[c]
15-10-5-5-P1	49	**426**	**643**	6404	**46**	498	1980	**89**
15-10-5-5-P2	46	**415**	**834**	7056	**45**	500	2418	**7**
15-10-5-5-P3	48	**380**	**503**	8410	**44**	497	2621	**61**
15-10-5-5-P4	48	**390**	**644**	8052	**45**	493	2492	**0**
15-10-5-5-P5	40	**399**	**2315**	8449	**39**	494	4159	**0**

[a]TotNumOfOrders: Total number of orders, [a]TotNumOfDists: Total number of distributions.
[b]TotInvOfDCs: Total amount of products in inventories of open DCs comprising all periods.
[c]TotInvOfRets: Total amount of products in inventories of retailers comprising all periods.

Regarding the results in Table 18, we see that when the value of the inventory cost weight factor for retailers changes to the 100 times of its original value, total amount of products in retailers' inventories decreases as expected. Besides, to decrease the

total amount of products in retailers' inventories, the total number of distributions made to the retailers and the total amount of products in open DCs' inventories increases. We also notice that as a balancing effect, total number of orders made by open DCs decreases a little bit.

Remember, in Section 3.3; we used SEH (shrinking and expanding) sub-procedures in the $findRoutes$ procedure to shrink or expand the routes taken as input for diversification purpose. In Fig. 3 we see the effect of the SEH sub-procedures. We run the $findRoutes$ procedure on a randomly generated 5-50-3-3 instance and graph the found best solutions with and without the SEH sub-procedure.

As Fig. 3 shows, the $findRoutes$ algorithm starts with a good initial solution of 86590.8 but improves this solution slightly without the SEH sub-procedure. On the other hand, the $findRoutes$ algorithm starts with a relatively bad initial solution of 90762.8 and improves this solution many times with SEH sub-procedure's help.

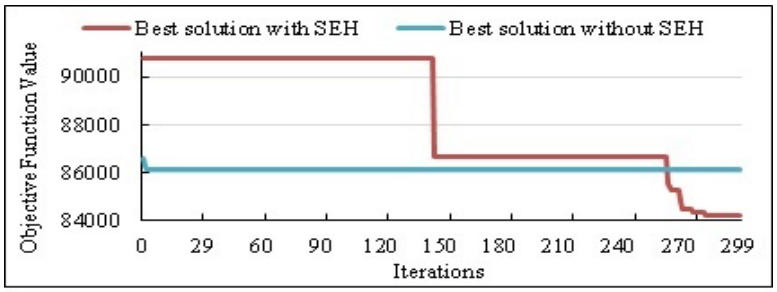


FIGURE 3. Effect of the SEH sub-procedures used in the $findRoutes$ procedure

To see the effect of fixed fleet strategies used in the $findAllRoutes$ procedure, we randomly create five new 5-50-3-3 instances and run the SHM on these instances five times with and without fixed fleet strategies. The results are seen in Table 19.

TABLE 19. Effect of fixed fleet finding strategies used in the $findAllRoutes$ procedure

Instance	Strategies are passive			Strategies are active		
	MinCost	Cost	CPU (s)	MinCost	Cost	CPU (s)
5-50-3-3-P6	193242.2	205648.1	0.098	170738.0	**175134.7**	0.153
5-50-3-3-P7	180165.0	192193.6	0.053	162587.4	**164062.5**	0.139
5-50-3-3-P8	178610.2	188135.7	0.041	165380.2	**170228.2**	0.077
5-50-3-3-P9	173728.4	184784.3	0.053	150471.2	**156236.2**	0.099
5-50-3-3-P10	157350.8	170610.8	0.053	144801.6	**148610.9**	0.105
Average	176619.3	188274.5	0.060	158795.7	**162854.5**	0.115

The second column show the best, the third column show the average distribution plus using vehicles at least once cost and the fourth column show the average computation time of the runs when the instances are solved without fixed fleet strategies. On the other hand, columns 5-7 show the same values when the instances are solved with fixed fleet strategies. We see that on the average, the average cost

of the active case is nearly 13.5% lower than the passive case with the expense of almost two times of computation time. Computation times are short so this drawback is acceptable.

We present the objective function value of the current and global best solutions for one of the runs of the 5-15-3-3-P1 instance in Fig. 4 to validate the intensification and diversification procedures of HHM. In this run, the intensification procedure is called three times and all of the phases (12 phases) are recorded. On the other hand, the diversification procedure is called two times and records are seen above 13. and 26. x axis labels.

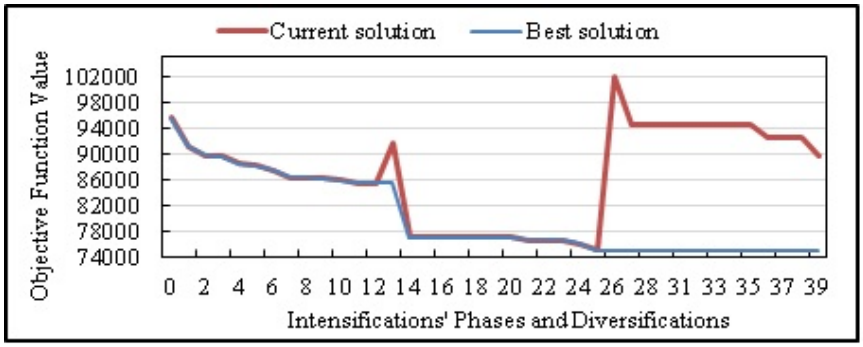


FIGURE 4. Analysis of intensifications' phases and diversifications

We see that current solutions are improved in all three intensification calls as supposed to be and the global best solution is improved in the first and second intensification calls. Both diversification calls create different solution configurations regarding open DC(s) and retailers assigned to the open DC(s). When we look at to the configurations of current solutions, we see that the initial solution includes two open DCs and three closed DCs. One of the open DCs is closed and one closed DC is opened in the first diversification call. In the second intensification call, one of the open DCs is closed and the global best solution is found with just only one open DC. In the second diversification call, one of the closed DCs is opened to intensify on a new solution configuration.

6. **Conclusions.** We present the MILRP with heterogeneous fleet for the first time as a comprehensive/generic approach to optimize many supply chain design scenarios including medicine/food delivery in humanitarian logistics, ammunition and other supplies distribution in the military logistics and cases in which leasing distribution centers are common, distribution to customers are made in different frequencies so the routing decisions change per period. MILRP with heterogeneous fleet considers a supply chain, which consists of multiple producers, potential distribution centers (DCs) with opening capacity levels, geographically scattered retailers, each of which has deterministic demand over a discrete planning horizon and heterogeneous fleet of vehicles. The goal is determining a set of DCs with their capacity levels to open (location decisions), assigning retailers to the opened DCs (allocation decisions) and for all periods finding the product quantities to be ordered by and distributed from DCs (inventory management decisions) and determining the fleet and routes (routing decisions) simultaneously to satisfy the demands of retailers with minimum cost.

We propose a mixed-integer linear programming model to describe the MILRP with heterogeneous fleet and strengthened it by adapting two valid inequalities proposed for ILRP. We see that only the very small-sized instances can be solved exactly within a reasonable computation time via ILOG CPLEX commercial solver after doing preliminary experiments. Therefore, we develop two heuristic methods (sequential and hybrid).

In the sequential heuristic method (SHM); we find the location-allocation decisions through a greedy algorithm first. Then, we solve our proposed inventory management model to find inventory management decisions exactly. Finally, we determine the vehicle routing decisions through a method based on tabu search metaheuristic. On the other hand, the hybrid heuristic method (HHM) consists of initialization, intensification and diversification steps. In the initialization step, we create the initial solution. Then, through the intensification process, which is the hybridization of simulated annealing and tabu search metaheuristics we improve this solution. Finally, with the diversification process, we change the location-allocation decisions of the best solution found by the intensification process and call the intensification process with this new solution. This loop is called for a pre-determined time and the best global solution is found. In order to create different solutions (having different location and allocation decisions), the location and allocation configurations tried before are stored in a list, which is updated by intensification and diversification processes.

Since there are no benchmark instances in the literature for MILRP with heterogeneous fleet, impact of two valid inequalities on both models (M1 model for MILRP and M2 model for inventory management decisions) and performance/solution quality of the developed heuristics and the ILOG CPLEX commercial solver are tested on randomly generated realistic instances. Results show that the two valid inequalities proposed for the M1 model and only the second valid inequality proposed for the M2 model are effective. In terms of solution quality, we can say that the quality of solutions found by SHM and HHM are acceptable for very small sized instances but we cannot guarantee this for the other sized instances due to the inability of making a comparison with solver. Nevertheless, we see that both heuristic methods provide important savings in acceptable run times compared to the commercial solver.

When we compare the heuristic methods in terms of solution time, we see that average computation time for SHM is substantially lower than HHM for small and medium sized instances, but it is the other way round for the large-sized instances. When we compare them in terms of solution quality, we see that HHM outperforms SHM in all sized instances.

The analyses made on medium-sized instances regarding the ordering, distribution and inventory cost weight factors, which are used in the M1 model show that all the results are compatible with the expected results. The presented benchmarks at the end of the results section prove some of our developed algorithms' effectiveness.

In conclusion, we can use both heuristic methods as a "what if" analysis tool for MILRP with heterogeneous fleet, which is a generic and frequently came across supply chain scenario. As future research, we are planning to improve our routing methodology first. We will then study the MILRP with two routing levels (routing added between producers and DCs) or the MILRP with time window constraints on distributions to retailers.

Compliance with Ethical Standards
Funding: We declare that there is no funding for this study.
Conflict of Interest: We declare that we have no conflict of interest.

REFERENCES

[1] D. Ambrosino and M. G. Scutella, Distribution network design: New problems and related models, *European Journal of Operational Research*, **165** (2005), 610–624.
[2] D. Blanchard, *Supply Chain Management Best Practices*, John Wiley & Sons, 2010.
[3] T. M. Cioppa and T. W. Lucas, Efficient nearly orthogonal and space-filling latin hypercubes, *Technometrics*, **49** (2007), 45–55.
[4] A. Ghorbani and M. R. A. Jokar, A hybrid imperialist competitive-simulated annealing algorithm for a multisource multi-product location-routing-inventory problem, *Computers & Industrial Engineering*, **101** (2016), 116–127.
[5] F. Glover and M. Laguna, Tabu search, In *Handbook of Combinatorial Optimization*, Vol. 3, Kluwer Acad. Publ., Boston, MA, 1998, 621–757.
[6] M. G. Granada and C. W. Silva, *Inventory Location Routing problem: A column Generation Approach*, PhD thesis, Uniandes, 2011.
[7] W. J. Guerrero, C. Prodhon, N. Velasco and C. A. Amaya, Hybrid heuristic for the inventory location-routing problem with deterministic demand, *International Journal of Production Economics*, **146** (2013), 359–370.
[8] W. J. Guerrero, C. Prodhon, N. Velasco and C.-A. Amaya, A relax-and-price heuristic for the inventory-location-routing problem, *International Transactions in Operational Research*, **22** (2015), 129–148.
[9] A. Hiassat, A. Diabat and I. Rahwan, A genetic algorithm approach for location-inventory-routing problem with perishable products, *Journal of Manufacturing Systems*, **42** (2017), 93–103.
[10] A. A. Javid and N. Azad, Incorporating location, routing and inventory decisions in supply chain network design, *Transportation Research Part E: Logistics and Transportation Review*, **46** (2010), 582–597.
[11] S. C. Liu and C. C. Lin, A heuristic method for the combined location routing and inventory problem, *The International Journal of Advanced Manufacturing Technology*, **26** (2005), 372–381.
[12] S. C. Liu and S. B. Lee, A two-phase heuristic method for the multi-depot location routing problem taking inventory control decisions into consideration, *The International Journal of Advanced Manufacturing Technology*, **22** (2003), 941–950.
[13] H. Ma and R. Davidrajuh, An iterative approach for distribution chain design in agile virtual environment, *Industrial Management & Data Systems*, 2005.
[14] N. Nekooghadirli, R. Tavakkoli-Moghaddam, V. R. Ghezavati and S. Javanmard , Solving a new bi-objective location-routing-inventory problem in a distribution network by meta-heuristics, *Computers & Industrial Engineering*, **76** (2014), 204–221.
[15] U. Pasha, A. Hoff and L. M. Hvattum, Simple heuristics for the multi-period fleet size and mix vehicle routing problem, *INFOR: Information Systems and Operational Research*, **54** (2016), 97–120.
[16] S. R. Sajjadi, M. Hamidi and S. H. Cheraghi, Multi-product capacitated location routing inventory problem, *International Journal of Modern Engineering*, **13** (2013).
[17] N. I. Saragih, N. Bahagia, I. Syabri, et al, A heuristic method for location-inventory-routing problem in a three-echelon supply chain system, *Computers & Industrial Engineering*, **127** (2019), 875–886.
[18] Z.-J. M. Shen and L. Qi, Incorporating inventory and routing costs in strategic location models, *European journal of operational research*, **179** (2007), 372–389.
[19] Y. Zhang, M. Qi, L. Miao and E. Liu, Hybrid metaheuristic solutions to inventory location routing problem, *Transportation Research Part E: Logistics and Transportation Review*, **70** (2014), 305–323.

Received January 2020; revised September 2020.

E-mail address: omer.arslan1@gazi.edu.tr
E-mail address: isleyens@gazi.edu.tr