Transportation Research Part E 70 (2014) 305–323

Contents lists available at ScienceDirect

Transportation Research Part E

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

Hybrid metaheuristic solutions to inventory location routing problem

Ying Zhang [a,b], Mingyao Qi [a,*], Lixin Miao [a], Erchao Liu [a]

[a] Research Center on Modern Logistics, Graduate School at Shenzhen, Tsinghua University, Shenzhen 518055, China
[b] Department of Industrial Engineering, Tsinghua University, Beijing 100084, China

ARTICLE INFO

Article history:
Received 21 April 2014
Received in revised form 25 July 2014
Accepted 26 July 2014

Keywords:
Supply chain
Inventory location routing
Inventory-routing problem
Location-routing problem
Metaheuristic
Simulated annealing

ABSTRACT

This paper considers a supply chain network with multiple depots and geographically dispersed customers, each of which faces non-constant demand over a discrete planning horizon. The goal is to determine a set of depots to open, the delivery quantities to customers per period and the sequence in which they are replenished by a vehicle fleet such that the total system-wide cost is minimized. To solve it, first we construct a mixed integer program, and then propose a hybrid metaheuristic consisting of initialization, intensification and post-optimization. Results show that the proposed heuristic is considerably efficient and effective for many classical instances.

1. Introduction

Fierce competition in today's global market forces companies to better design and manage their supply chain networks, which usually involves facility location, inventory and transportation decisions. The coordination and integration of these components are critical issues as they affect the long term operating cost and the quality of customer service.

This paper studies a two-echelon supply chain network planning problem. We refer it as the inventory location routing problem (ILRP), integrating strategic, tactical and operational decisions. It is essential to balance short-term decisions with longer term thinking to satisfy future demands at minimum cost (Guerrero et al., 2013). ILRP consists of three subproblems: location allocation problem (LAP), inventory control problem (ICP) and vehicle routing problem (VRP). LAP is to choose the best locations for depots from a given set of candidate sites to minimize the sum of opening cost and the cost of assigning customers to the depots. ICP is to determine the delivery quantity and frequency of each customer to satisfy its demand guaranteeing that no stock-out occurs. VRP is to determine the optimal delivery routes from a given depot to some geographically dispersed customers. Since daily routing highly depends on the delivery quantity and frequency, and the inventory-routing decisions greatly affect the strategic location decisions, these three problems are highly correlated with each other.

In the literature, some strategic location models incorporate estimated routing and/or inventory costs, but no accurate routing or inventory decisions. Nozick and Turnquist (1998) tried to estimate the inventory costs and include them within a fixed-charge facility location model. The cost of safety stock was approximated as a linear function of the number of depots, while transportation cost was represented by straight-line or radial trips. Max Shen and Qi (2007) proposed a supply chain

* Corresponding author. Tel.: +86 755 26036130; fax: +86 755 26036005.
E-mail addresses: yingzhang8996@gmail.com (Y. Zhang), qimy@sz.tsinghua.edu.cn (M. Qi), lxmiao@tsinghua.edu.cn (L. Miao), liuerchao1989@foxmail.com (E. Liu).

http://dx.doi.org/10.1016/j.tre.2014.07.010
1366-5545/

design model which considered the impacts of facility location decisions on inventory costs and distribution costs. But the specific routing decisions were ignored, for the corresponding costs were estimated by a continuous approximation approach. Miranda and Garrido (2008) presented an inventory location problem, which was to locate a set of warehouses to serve spatially distributed clients, taking safety stock and holding costs into account. Rennemo et al. (2014) considered the location of local distribution facilities, while including the last mile distribution decisions but no stock of emergency supplies, for a disaster response problem.

Some location-routing problems (LRP) include inventory costs in the objectives but not inventory decisions. Liu and Lee (2003) and Liu and Lin (2005) considered the single-product multi-depot LRP. They adopted an order-up-to continuous review model to represent inventory costs just with location and routing decisions. The former proposed a two-phase heuristic method, while the latter presented a hybrid heuristic combining tabu search (TS) and simulated annealing (SA). For small-sized problems with 3 depots and 6 customers, this method is comparative than the enumeration search. Ma and Davidrajuh (2005) considered a system with a central depot, potential wholesalers and retailers. They proposed an iterative approach where the configuration of distribution chain was constantly optimized by transferring data between the strategic model (aimed at selecting wholesalers) and tactical model, in which the demand of each retailer was determined using a (s, Q) policy. Never the global perspective can be guaranteed.

Inventory-routing problem (IRP), which considers distribution systems with one depot and multiple geographically dispersed retailers, has got much attention in the last thirty years. With inventory decisions inside, this problem does not contain location decisions. For example, Cha et al. (2008) and Huang and Lin (2010) both considered the one-warehouse, n-retailer system. Their objectives were to determine the optimal replenishment policy and delivery scheduling, while no location decisions inside. A detailed review can be referred to Moin and Salhi (2006) and Andersson et al. (2010).

Lately, some authors begin to simultaneously optimize location, inventory and routing decisions. Javid and Azad (2010) are the pioneers. They considered the single period decision in a stochastic supply chain system, where each customer's demand followed a normal distribution and each DC maintained a certain amount of safety stock and was assumed to follow a (Q, R) inventory policy. The proposed methodology was based on TS and SA. Mete and Zabinsky (2010) developed a stochastic programming for disaster preparedness to determine an optimal policy of warehouse selection and inventory levels in the first stage, as well as a collection of resource decisions on transportation plans in the second stage. But it is mainly about the stock of medical supply rather than inventory policy. Reza Sajjadi and Hossein Cheraghi (2011) considered a multi-product network and adopted the fixed order interval policy.

Since the single period model may not reflect the long-term planning (Moin and Salhi, 2006), a few studies further focus on the multi-period version. Planning horizon is divided into elementary time slots (called "periods" or "days", Prodhon and Prins, 2014). Different demands and so different delivery amount (may be zero) can be specified at each period. Ambrosino and Grazia Scutellà (2005) proposed two formulations both in the static scenario (single period) and dynamic scenario (multi period, deterministic). They solved the single period case of small size (less than 13 depots and 95 retailers) with CPLEX 7.0, while no methodologies and computational results were given for the multi-period case. To our best knowledge, Guerrero et al. (2013) for the first time presented an algorithm for a multi-period ILRP with deterministic demand. Three layers were considered: one factory, a set of potential depots, and a set of retailers. Several decisions were taken simultaneously: location decisions of depots, allocation decisions of retailers to depots, inventory decisions at both depots and retailers and multi-period routing decisions to satisfy retailers' demand. The authors proposed a matheuristic to solve location-allocation and inventory decisions (neglecting the routing construction) using a commercial solver.

Because of the generalized complexity of inventory-routing category, the literatures introduced above all adopted some computationally efficient policies for certain problems, such as fixed-partition policy, the power-of-two (POT) policy and order-up-to policy. Anily and Federgruen (1990) adopted the fixed-partition policy for a demands retailers-specific system. This policy partitions the retailers to a set of regions. Each time one of the retailers in a given region receives a delivery, this delivery is made by a vehicle who visits other retailers in this region as well. Viswanathan and Mathur (1997) adopted a nested stationary POT policy for a multi-echelon and multi-product system. Under a POT policy, replenishments occur every $T = 2^k T_B$ periods, where T_B is some base planning period. A stationary policy is characterized by constant replenishments intervals, while in a nested policy, for two customers i and j, if $T_i > T_j$, then i must be a multiple of j. Bertazzi et al. (2002) presented a deterministic order-up-to policy for a multi-product system. This policy ensures that, every time a retailer is visited, the quantity delivered is such that the maximum level of the inventory is reached. Those above inventory strategies heavily dictate the structure of the solution produced (Zachariadis et al., 2009): following a fixed partition policy, customer subgroups are bounded to be visited independently of each other; under a POT and nested strategies, customers are forced to be replenished at stationary intervals. Not limited to these limitations, an alternative replenishment policy allows flexible reorder point and order quantity for different customers, which was adopted by Zachariadis et al. (2009) and Qin et al. (2014) in IRP area.

This paper considers an integrated inventory location routing problem and adopts the above flexible replenishment policy – instead of just determining the frequency or size of the replenishment services, we aim to construct long-term replenishment strategies (i.e., inventory rules and routing patterns) over a finite planning horizon during which all customers must be visited and served. The goal is to minimize long-run average system-wide location, inventory and routing costs. Moreover, to our best knowledge, at present till the research integrating these respects is inadequate still. Since the presented problem reduces to the well-known VRP for each opened depot and each period, it is NP-hard. To optimally solve small-sized instances, the problem is cast as a mixed-integer linear programming model. For larger ILRP instances, a hybrid

metaheuristic integrating initialization, intensification and post-optimization is proposed, which can get "good solutions" in acceptable time (Barreto et al., 2007).

The major difference between this study and Guerrero et al. (2013) is in the methodologies, especially in tackling with inventory-routing decisions. Suppose initial location-allocation decisions are fixed, the previous research handles the inventory and routing decisions in a separate way, while in this paper we tackle it integratedly. They formulate a dynamic lot-sizing problem to determine the optimal quantities to stock at the depots and retailers, with estimated distribution costs, and solve it with a MIP solver. As a result, the size of the targeted instances is strictly restricted, with an average computation time of 8471.8 s to solve a randomly generated instance with 5 depots, 15 retailers and five periods. While this paper designs efficient operators which embedded with small subproblems. Therefore, the size of the instances targeted in this paper can be up to 25 depots, 300 customers and seven periods, with no obvious increase in CPU time.

The main contribution of this paper is threefold: firstly, a new mixed integer formulation for the multi period ILRP is first constructed. Secondly a smart representation of inventory cost is proposed which can decrease computational complexity to a large extent. Thirdly an efficient hybrid metaheuristic is designed for the large-scale problems.

The remainder of this paper is outlined as follows: in next section, a novel mathematical model is given. Section 3 describes a low complexity method to calculate inventory cost. Then in Section 4, the presented solution methodology is provided in detail, followed by the computational results in Section 5. Some conclusions and future research are listed in the last section.

2. Problem definition

This paper deals with a two-echelon supply chain network consisting of multiple potential capacitated depots and a set of ordered customers. Each customer faces deterministic but period-variable demand over a given planning horizon. Inventories are kept at the customers not at those depots. In other words, the depot serves as coordinators of the replenishment process and they act as "break-bulks" (Anily and Federgruen, 1990) or transshipment points from where all vehicles start and return to. A homogeneous fleet of capacitated vehicles carry a single product to satisfy customers' demands. Both strategic and tactical/operational decisions are considered. The objective is to determine the schedules to coordinate location, allocation, inventory and transportation routing operations so that the overall cost is minimized.

First we give the generalized graph representation. Define I to be the set of customers, indexed by i, and J to be the set of candidate depots, indexed by j. For ease of notation, we also use I and J to indicate the cardinalities of the sets. This problem is defined on a weighted and directed graph $G = (V, A)$ where the set of nodes $V = I \cup J$ and the set of arcs is $A = \{(i, j), i, j \in V, i \neq j\}$. Each pair of location $(i, j) \in A$ is associated with a travelling cost c_{ij}. With each customer $i \in I$ is associated with a deterministic non-constant demand $d_{it}, \forall t \in H$, where H is a set of discrete and finite planning horizon, and a unit inventory holding cost h_i. At each period $t \in H$, each customer can be visited at most once – by any of the K available vehicles. The total amount delivered to each customer over the planning horizon must be equal to its total demand $D_i = \sum_{t \in H} d_{it}$. Each depot $j \in J$ has an open cost f_j and storage capacity C_j. Each used vehicle $k \in K$ with loading capacity Q_k travels from a specific depot to visit customers and returns to that depot. Backlogging or stock-out is not allowed.

To provide a mathematical formulation, we introduce the following notation: let the decision variable $y_j = 1$ if $j \in J$ is opened; $z_{ij} = 1$ if customer i is assigned to a depot based at j, and 0 otherwise; $x_{ijkt} = 1$ if node j is visited immediately after node i on period t by vehicle k. Variable q_{ikt} represents the product quantity delivered to customer i, on period t, by vehicle k. We also introduce auxiliary variables w_{itp} to represent the quantity delivered to customer i in period p to satisfy its demand in period t. Then the ILRP model can be stated as follows:

$$(\textbf{ILRP})\ \min \sum_{j \in J} f_j y_j + \sum_{i \in I} h_i \sum_{t \in H} \left(\frac{1}{2} d_{it} + \sum_{p \in H, p < t} w_{itp}(t - p) + \sum_{p \in H, p > t} w_{itp}(t - p + H) \right) + \sum_{i \in V} \sum_{j \in V} \sum_{t \in H} \sum_{k \in K} c_{ij} x_{ijkt} \tag{1}$$

Subject to

$$\sum_{j \in V} x_{ijkt} - \sum_{j \in V} x_{jikt} = 0 \quad \forall i \in V,\ \forall k \in K,\ \forall t \in H \tag{2}$$

$$\sum_{j \in V} \sum_{k \in K} x_{ijkt} \leqslant 1 \quad \forall t \in H,\ \forall i \in I \tag{3}$$

$$\sum_{j \in V} \sum_{k \in K} x_{jikt} \leqslant 1 \quad \forall t \in H,\ \forall i \in I \tag{4}$$

$$\sum_{i \in I} \sum_{j \in J} x_{ijkt} \leqslant 1 \quad \forall k \in K,\ \forall t \in H \tag{5}$$

$$x_{ijkt} = 0 \quad \forall i,\ j \in J,\ \forall k \in K,\ \forall t \in H,\ i \neq j \tag{6}$$

$$\sum_{i \in I} q_{ikt} \leqslant Q_k \quad \forall k \in K,\ \forall t \in H \tag{7}$$

$$\sum_{i \in S} \sum_{j \in S} x_{ijkt} \leqslant |S| - 1 \quad \forall k \in K,\ \forall t \in H,\ \forall S \subseteq I \tag{8}$$

$$\sum_{j\in J} z_{ij} = 1 \quad \forall i \in I \tag{9}$$

$$z_{ij} \leqslant y_j \quad \forall i \in I,\ \forall j \in J \tag{10}$$

$$\sum_{i\in I}\left(z_{ij}\sum_{t\in H} d_{it}\right) \leqslant C_j \quad \forall j \in J \tag{11}$$

$$\sum_{u\in I} x_{ujkt} + \sum_{u\in V\setminus\{i\}} x_{iukt} \leqslant 1 + z_{ij} \quad \forall i \in I,\ \forall j \in J,\ \forall k \in K,\ \forall t \in H \tag{12}$$

$$\sum_{i\in I}\sum_{k\in K}\sum_{t\in H} x_{jikt} \geqslant y_j \quad \forall j \in J \tag{13}$$

$$\sum_{i\in I} x_{jikt} \leqslant y_j \quad \forall j \in J,\ \forall k \in K,\ \forall t \in H \tag{14}$$

$$\sum_{p\in H} w_{itp} = d_{it} \quad \forall i \in I,\ \forall t \in H \tag{15}$$

$$\sum_{t\in H} w_{itp} = \sum_{k\in K} q_{ikp} \quad \forall i \in I,\ \forall p \in H \tag{16}$$

$$q_{ikt} \leqslant M\sum_{j\in V} x_{ijkt} \quad \forall i \in I,\ \forall t \in H,\ \forall k \in K \tag{17}$$

$$\sum_{j\in V} x_{ijkt} \leqslant M q_{ikt} \quad \forall i \in I,\ \forall t \in H,\ \forall k \in K \tag{18}$$

$$x_{ijkt} \in \{0,1\} \quad \forall i \in I,\ \forall j \in J,\ \forall t \in H,\ \forall k \in K \tag{19}$$

$$y_j \in \{0,1\} \quad \forall j \in J \tag{20}$$

$$z_{ij} \in \{0,1\} \quad \forall i \in I,\ \forall j \in J \tag{21}$$

$$q_{ikt} \leqslant min\left\{Q_k, \sum_{p\in H} d_{ip}\right\} \quad \forall i \in I,\ \forall j \in J,\ \forall k \in K \tag{22}$$

$$w_{itp} \leqslant d_{ip} \quad \forall i \in I,\ \forall t, p \in H \tag{23}$$

The objective function Eq. (1) is the sum of the opening costs, holding costs at customers and the total routing costs over the planning horizon.

Concerning distribution, constraints (2)–(8) force a feasible routing. First, traditional vehicle flow conservation are presented in Eqs. (2)–(4). Constraints (2) ensure that when a vehicle visits a node, it does also depart from it. Each customer is allowed to be served by at most one vehicle in a period, which is guaranteed by (3) and (4). Note that constraints (5) force each vehicle to perform one route per period at the most. (6) prevent the vehicles to travel from a depot to another, which narrow the solution space. Constraints (7) are the capacity constraints of the vehicles. Constraints (8) are standard subtour elimination constraints guaranteeing that the depot is present in every generated route. If we introduce auxiliary variables $U_{ikt} \in \{0, 1, \cdots, I-1\}$ for $\forall i \in I, \forall k \in K, \forall t \in H$, constraints (8) can be expressed as follows

$$U_{ikt} - U_{jkt} + Ix_{ijkt} \leqslant I - 1 \quad \forall i, j \in I,\ \forall k \in K,\ \forall t \in H \tag{24}$$

Constraints (9)–(14) are location constraints which connect x, y and z. Each customer must be allocated to a single opened depot as stated by Eqs. (9) and (10). Capacity of depots is guaranteed by (11). The set of Eq. (12) state that a customer i can be linked to a depot j only if i is assigned to depot j ($z_{ij} = 1$). A vehicle is routed from a depot if and only if that depot is opened, which is guaranteed by (13) and (14).

Constraints (15)–(18) are those inventory related constraints. Constraints (15) force the satisfaction of the demand at each customer, while (16) ensures that sum of the amount of products that responsible for each customer on each period equals its delivered quantity. Constraints (17) and (18) guarantee that if a customer is replenished on period t by vehicle k, it must be visited accordingly.

Finally (19)–(23) state the nature of the decisions/auxiliary variables.

Note that dealing with the vehicle capacity constraints, Zachariadis et al. (2009) and Qin et al. (2014) both introduced nonlinear terms. Meanwhile, the w-decisions in Guerrero et al. (2013) have five subscripts, with a number of $J{\cdot}I{\cdot}H{\cdot}H{\cdot}K$ variables, which makes it more difficult to solve. Besides, the initial inventory associated with each node is actually unnecessary for a multi-period problem.

3. Inventory cost calculation

In this section, we will present our new novel method to calculate the inventory cost, where a theorem will be shown.

Once a new replenishment policy is scheduled, the inventory cost of the corresponding customer needs to be recalculated. Qin et al. (2014) proposed a somewhat complicated method to calculate inventory cost. For each customer, three arrays are introduced to find out the exact deliveries, i.e., auxiliary variables w_{itp} defined in the formulation, which satisfied the demand

on each period. To determine the value of these arrays, too many loops are executed over the horizon. In our test, we find that w_{itp} need not to be explicitly determined. The following theorem shows a lower complexity algorithm.

Theorem 1. *In a multi-period decision case, for each customer, if the total amount of goods it received equals its total demand over each single horizon, the inventory strategy is feasible for multiple horizons.*

Proof. Assume that d_{it} is the demand of customer i on period t, q_{it} represents the quantity delivered to customer i on period t. We need to prove that if $\sum_{t\in H} d_{it} = \sum_{t\in H} q_{it}$, then stock out never occurs for arbitrary customer $i \in C$ over the planning horizon. Note that for a multi-period case, at the beginning of the horizon there might be some initial inventory remained from the horizon before.

Let $b_{it} = q_{it} - d_{it}$, $\forall i \in C$, $\forall t \in H$, as the difference between the delivery and demand of customer i at period t. For each $t_0 \in H$, we can define s_{it} as the accumulated difference:

$$s_{it}(t_0) = \begin{cases} \sum_{\tau=t_0}^{t} b_{i\tau} & t \geqslant t_0 \\ \sum_{\tau=t_0}^{H} b_{i\tau} + \sum_{\tau=1}^{t} b_{i\tau} & t < t_0 \end{cases} \tag{25}$$

From the definition above, we have $s_{it} = s_{i,t-1} + b_{it}$. Then Theorem 1 shows that there must be a value of t_0, named t_0^* such that for all $t \in H$, we have $s_{it}(t_0^*) \geqslant 0$.

Suppose, for a contradiction, that for all $t_0 \in H$, there exists a customer to which stock out happens ($s_{it}(t_0) < 0$). Without loss of generality, we take $t_0 = t + 1$. From the definition

$$s_{it}(t_0) = \sum_{\tau=t_0}^{H} b_{i\tau} + \sum_{\tau=1}^{t} b_{i\tau} = \sum_{\tau=t_0}^{H} b_{i\tau} + \sum_{\tau=1}^{t_0-1} b_{i\tau} = \sum_{\tau\in H} b_{i\tau} < 0 \tag{26}$$

But, as introduced before, $\sum_{\tau\in H} b_{i\tau} = \sum_{\tau\in H} q_{i\tau} - \sum_{\tau\in H} t_{i\tau} = 0$, leading to a contradiction. □

According to Theorem 1, if the specific t_0^* is found, then the inventory cost of customer i is

$$CI_i = h_i \sum_{t\in H} \frac{1}{2} d_{it} + h_i \sum_{t\in H} s_{it}(t_0^*) \tag{27}$$

The first part of (27) is the average holding cost, while the second part represents the accumulated inventory cost. That means when the quantity delivered on each period is determined, the inventory cost of this customer is known, without dependence on w_{itp}. The worst case complexity is $O(H^2)$.

Table 1 shows a simple example with a planning horizon of five periods. The initial inventory is 10. For this customer, the demand and delivery quantity on each period are listed in the second row and third row, respectively. Fig. 1 provides the diagram of the inventory level against period t. According to the analysis above, $t_0 = 3$ can ensure that $s_{it}(3) \geqslant 0$ for all $t = \{1, 2, 3, 4, 5\}$. So $t_0 = 3$ is the "*start time*" to calculate the inventory cost under this strategy. Notice that the inventory level reaches the same level of 20 after each replenishment; however, this is not a necessity but a coincidence.

If $h = 1$, according to the theorem, $CI_i = h_i \sum_{t\in H} \frac{1}{2} d_{it} + h_i \sum_{t\in H} s_{it} = 60 \times \frac{1}{2} \times 1 + 1 \times (10 + 0 + 10 + 0 + 10) = 60$, the area of surface under the polyline is $\frac{1}{2} \times (20 \times 2 + 30 \times 1 + 20 \times 1 + 30 \times 1) = 60$. This validates the effectiveness of Eq. (27).

4. Hybrid metaheuristic

The studied ILRP composes of LAP, VRP and it also aims to determine the optimal timing and size of customer replenishments. Therefore, to produce high quantity solutions with reasonable computation times, one's interest must focus on heuristic and metaheuristic approaches. The proposed framework is based on this kind of mentality, trying to exchange information among solution spaces.

Fig. 2 presents a general flowchart of the proposed algorithm. In the following, we will first propose the method to generate an initial solution. Through initialization, several decisions must be determined: location-allocation, which is the output of the (CFLP) model presented in Section 4.1, and inventory-routing, which is initialized by the Solomon I1 algorithm

Table 1
The inventory strategy.

Period	1	2	3	4	5
Demand	10	10	10	20	10
Delivery	10	0	20	10	20
Difference	0	−10	10	−10	10
$s_{it}(3)$	10	0	10	0	10

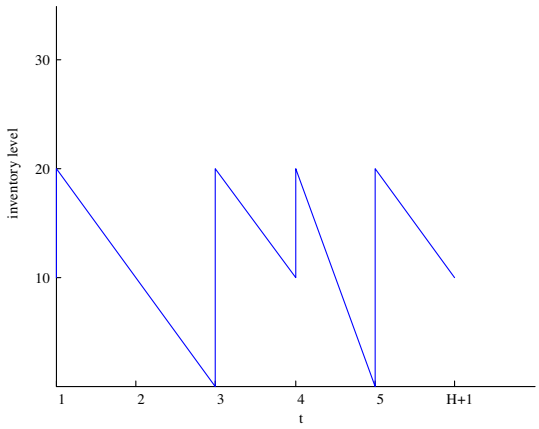


Fig. 1. A simple example to show the inventory cost calculation.

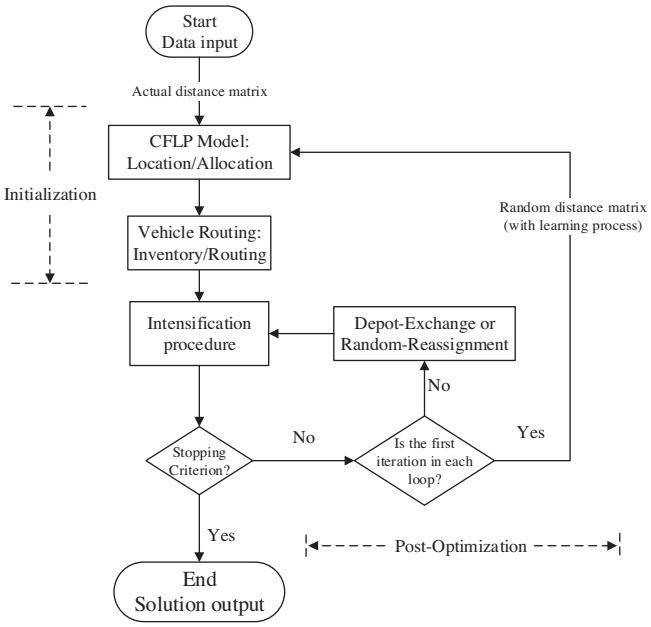


Fig. 2. Flowchart of solution procedure for ILRP.

and optimized by the *Vehicle Routing* procedure, as introduced in Section 4.2. After that, an in-depth description of five local search operators is provided, followed by the presentation of intensification procedure, which is implemented by a SA framework (so we will rename the proposed algorithm as SA-Hyb-ILRP). Finally, we introduce the disturbance mechanisms in the post-optimization procedure to expand the search scope and avoid trapping in local optimum. Post-optimization is realized by randomly initialization or adding some diversity to current solution, and then calling the intensification procedure to impose local optimization.

Table 2 gives a main overview of proposed algorithm from the global perspective. C_0 is the real distance matrix to be input to the initialization procedure. To explain the relationship between the decisions and algorithms in these procedures, Section 4.1 is the initialization process, giving initial decisions on location, inventory and routing. Section 4.2 improves the routing decisions for each given depot and given period. Section 4.3 is an intensification procedure which can explore different location, inventory and routing decisions simultaneously, or part of them. With all these done, the solution can be further diversified by some "big change" operators to seek in wider space, which is explained in Section 4.4. The detailed descriptions will be shown in the subsequent sections.

4.1. Initialization

To determine the inventory scheme and routing schedule, we need to find out the depot location and customer–depot allocation first. The Capacitated Fixed-charge Location problem (CFLP) is modeled to decide how many depots should be built and where they should be located (Daskin, 1995). In the first stage of our initialization process, CFLP model is adopted to determine the location-allocation decisions. Here we just ignore the inventory cost and routing cost, also, we take the value of c_{ij} as the direct distance instead of demand-weighted distance:

$$(\mathbf{CFLP})\min = \sum_{j\in J} f_j y_j + \sum_{i\in I}\sum_{j\in J} c_{ij} z_{ij} \tag{28}$$

Subject to

$$\sum_{j\in J} z_{ij} = 1 \quad \forall i \in I \tag{29}$$

$$z_{ij} \leqslant y_j \quad \forall i \in I, \forall j \in J \tag{30}$$

$$\sum_{i\in I} (z_{ij} D_i) \leqslant C_j \quad \forall j \in J \tag{31}$$

$$y_j \in \{0,1\} \quad \forall j \in J \tag{32}$$

$$z_{ij} \in \{0,1\} \quad \forall i \in I, \forall j \in J \tag{33}$$

$D_i = \sum_{t\in H} d_{it}$ is the total demand of customer $i \in I$ over the given planning horizon. The other variables and notations are the same as the model presented in Section 2. By relaxing capacity constraints (31), this problem can be solved using a Lagrangian relaxation approach. A binary knapsack problem and transportation problem must be solved in solving the relaxation sub-problems. The details are ignored (see Daskin, 1995).

After determining the assignment, we can solve a VRP for each opened depot separately, in the second stage of initialization process. Each period over the horizon, we just let the quantity delivered to the customers equal to the corresponding demand on that period. Then each opened depot and each period $t \in H$ is associated with a set of routes. Each route is initialized by an I1 insertion algorithm proposed in Solomon (1987), and then improved by applying a *Vehicle Routing* procedure, which will be introduced in next section. This initial solution is then transferred to the intensification phase (Section 4.3).

4.2. Vehicle routing

Given an open depot j and the associated period t in which replenishment plan has been affected, the routing schedules could be further improved by the following granular tabu search (GTS) heuristic, serving every customer i for which $q_{it} > 0$. GTS begins from an initial route, obtained either by the I1 algorithm or resulting from local search operators in the intensification procedure.

The vehicle routing optimization process based on the GTS defined below, without changing the set of open depots and without reassigning a customer from a depot to another, hence is exactly the single depot VRP. In this section for ease of notation, we denote n_c as the number of customers assigned to the depot considered.

GTS introduced by Toth and Vigo (2003) is a very promising concept and has yielded excellent results on the VRP. The main idea behind GTS stems from the observation that the longer edges have only a small likelihood of belonging to an optimal solution. Therefore, by eliminating all edges whose length exceeds a *granularity threshold* v, GTS searches a smaller neighborhood, it is faster than the original TS. They suggested setting $v = \beta\bar{c}$, where β is called a *sparsification parameter*, and $\bar{c}$ is the average edge length of a solution obtained by a fast heuristic. In this paper, we use the well-known Solomon I1 insertion algorithm. Every time we call this *Vehicle Routing* procedure, the depot and associated customers may be different, we execute I1 to recalculate $\bar{c}$. β is initialized to 1.25 according to Toth and Vigo.

Table 2
Pseudocode of the main algorithm.

Procedure: main algorithm
1. Get initial solution $S_0 \leftarrow$ *initialization* (C_0)
2. $S^* \leftarrow$ *intensification* (S_0)
3. $S_{new} \leftarrow$ post-optimization (S^*)
4. **if**(Cost$(S_{new}) < Cost(S^*)$)
5. $S^* = S_{new}$
6. end if
7. $S^* \leftarrow$ *intensification* (S^*)
8. Return S^*

In each iteration inside the loop, some new edges may be generated by searching the neighborhood. Here we only consider the edges that belong to the restricted edge set $E(\upsilon) = \{(i,j)A|c_{ij} \leqslant \upsilon\} \cup I$, where I is a set of *important* edges defined as those incident to the depot, and those belonging to the best solution found and to the incumbent. Unlike the GTS proposed by Toth and Vigo (2003), we consider a different structure of neighborhoods, which can operated both inter- and intra-routes and can search a much wider space, not just limited to small number of arc exchanges and customer movements.

The neighborhoods are found with four strategies named *relocate*, *2-opt*, *exchange* and *cross-exchange*. These neighborhoods try to improve current solution by exchanging some nodes or edges, or relocating them to some other positions, or combining two routes: "*relocate*" operator is to relocate a customer from its current position to another position; "*2-opt*" operator replaces two arcs by two other arcs; "*exchange*" operator swaps simultaneously two different customers; "*cross-exchange*" operator aimed at swapping multi-nodes in different routes. They share the same probability to be selected. The detailed explanations and demonstrations of these neighborhoods may be referred to Qi et al. (2013). At each iteration, a neighborhood structure is randomly selected, and the whole neighbors with at least one edge belonging to the set $E(\upsilon)$ consist of the candidate list. The best move with non-tabu new edges is performed even if it is non-improving. Once a move is executed, those deleted edges are considered tabu during a small constant tenure, which is fixed at 7 in our computational tests.

Also, to diversify the search, β is dynamically increased to 1.75, whenever the current best solution is not improved after $15n_c$ iterations. Then the best solution found is set as the incumbent to perform n_c iterations. Then, β is reset to 1.25 and the search continues. All evaluated moves must be feasible, and must not be tabu unless the best candidate is superior to the best solution ever. If the incumbent has been set as the best solution found for TS_m times, the procedure stops. In our computational tests, $TS_m = 80$ and the value of other parameters are the same as Toth and Vigo (2003), as we already show above. An overview of the procedure is provided in Table 3.

4.3. Intensification

The proposed algorithm framework explores the solution space by employing moves from one solution to another. To perform these moves, we have designed five local search operators which jointly tackle the location and inventory and routing characteristics of the problem. The first three operators change the replenishment strategies by moving some delivery quantities from one period to others. The last two operators change the allocation properties.

4.3.1. Local search operators

Denote $r(i,t)$ as the route which serves customer i on period t and $R(j,t)$ as the set of routes that depart from depot j on period t. A_i is the assigned depot of customer i. Obviously $r(i,t) \in R(A_i,t)$. For convenience, we will use q_t instead of q_{it} for a certain customer $i \in I$, unless otherwise stated.

Table 3
Pseudocode of the granular tabu search.

Procedure: vehicle routing (S_0)

1. Initial $TS_m = 80$, $S^* = S_0$, stopCnt = 0, sameCnt = 0, diversityCnt = 0, $\beta = 1.25$
2. Set flag = true //start with the incumbent
3. Use I1 algorithm to determine the value of $\bar{c}$
4. **while** (stopCnt < TS_m)
5. Update the restricted edge set $E(\upsilon)$
6. Randomly select RN from {relocate, 2-opt, exchange, cross-exchange}
7. Get the best candidate $S_{new} \leftarrow RN(S_0)$, whose newly generated edges are non-tabu and belong to $E(\upsilon)$
8. Declare the deleted edges with the implement move tabu for seven iterations
9. Update $S_0 = S_{new}$
10. **if**(Cost(S_0) < $Cost(S^*)$)
11. Update current best solution $S^* = S_0$
12. sameCnt = 0
13. else
14. sameCnt = sameCnt + 1
15. end if
16. **if**(sameCnt $\geqslant 15n_c$ || flag == false)
17. $S_0 = S^*$, $\beta = 1.75$, flag = false //start with the best solution found
18. diversityCnt = diversityCnt + 1
19. end if
20. **if**(diversityCnt $\geqslant n_c$)
21. sameCnt = 0, diversityCnt = 0, $\beta = 1.25$, flag = true //start with the incumbent
22. stopCnt = stopCnt + 1
23. end if
24. end while
25. return S^*

4.3.1.1. Shift-Delivery operator. By this operator, for arbitrary customer $i \in I$, two periods $t_1(q_{t1} > 0)$ and t_2 are randomly selected, and the delivery q_{t1} is shifted to period t_2. If customer i is not served on period t_2 ($q_{t2} = 0$), then i is selected to be inserted into a route of $R(A_i, t_2)$ according to the minimum insertion cost criterion, with all or part of q_{t1} shifted to that route. Otherwise, if customer i is already served on t_2 ($q_{t2} > 0$) and route $r(i, t_2)$ is not fully loaded, then all or part of q_{t1} will be added to q_{t2} .

Suppose that the route which customer i will be inserted into has available capacity q_m, then the shifted quantity on both cases is $\Delta q = min\{q_{t1}, q_m\}$. If $q_m \geqslant q_{t1}$, then $q_{t1} - \Delta q = 0$ and customer i is removed from $r(i, t_1)$, leading to some decrease of routing cost.

This operator is named as $Shift(i, t_1, t_2)$. Base on the replenishment plan of Fig. 1, we are committed to shift the delivery on period $t_1 = 5$ ($q_{t1} = 20$) to another period. Suppose the route which customer i is to be inserted has available capacity $q_m = 10$, so $\Delta q = \min\{q_{t1}, q_m\} = 10$. Fig. 3 presents two cases of this operation.

For one example, if $t_2 = 2$, since the customer is not served on t_2, so we insert it into a route according to the minimum insertion cost criterion, with quantity 10 of q_{t1} transferred to that route; for the case that $t_2 = 1$, since the customer is already served on t_2, we then transfer quantity 10 from q_{t1} to q_{t2}. After employing this operator, the inventory costs in two cases changed to $CI_a = 80$ and $CI_b = 100$, respectively.

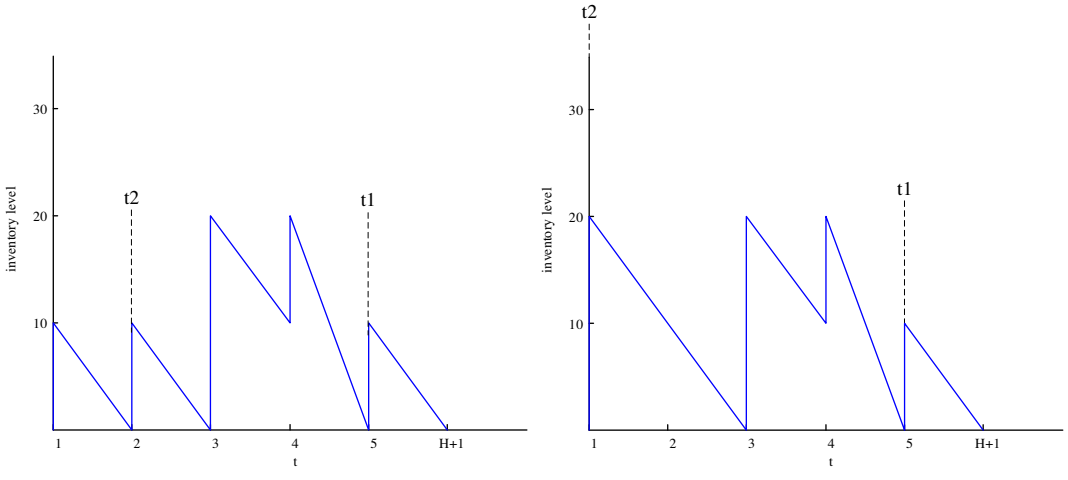
4.3.1.2. Two-Shift-Delivery operator. While being similar to Shift-Delivery operator, this operator can consider a couple of customers which share the same depot. This idea is inspired by the ejection chain operator in the VRP scope (Rego, 1998). If we are interested in shift the delivery of customer i_1 from period t_1 to t_2, while the routes in $R(A_{i1}, t_2)$ do not have enough capacity to "accommodate" it, we can first shift some quantity of another customer i_2 from period t_2 to t_3 in advance to "reserve" some capacity. Or we can view this operator as a combination of two Shift-Delivery operators. By our test, we find that this operator is more efficient, with higher possibility to find a better solution.

Fig. 4 presents the principle of this operator. In detail, Fig. 4a and b illustrate the initial replenishment strategy of two customers. Now we want to shift the delivery quantity of customer i_1 from t_1 to t_2, where $t_1 = 4, t_2 = 1, q_{i_1 t_1} = 10, q_{i_1 t_2} = 10$. But route $r(i_1, t_2)$ do not have enough capacity to accommodate some more goods. We observe that the delivery quantity of another customer i_2 on route $r(i_1, t_2)$ could move to period t_3, where $t_3 = 2, q_{i_2 t_2} = 10, q_{i_2 t_3} = 10$. So we apply Shift-Delivery operator twice on these two customers: first $Shift(i_2, t_2, t_3)$ and then $Shift(i_1, t_1, t_2)$.

Or in other words, Two-Shift-Delivery operator can be defined as: $TwoShift(i_1, i_2, t_1, t_2, t_3) = Shift(i_2, t_2, t_3) \cup Shift(i_1, t_1, t_2)$. After this operation, two customers have their replenishment plan changed, in the same time, three delivery routes also have been changed.

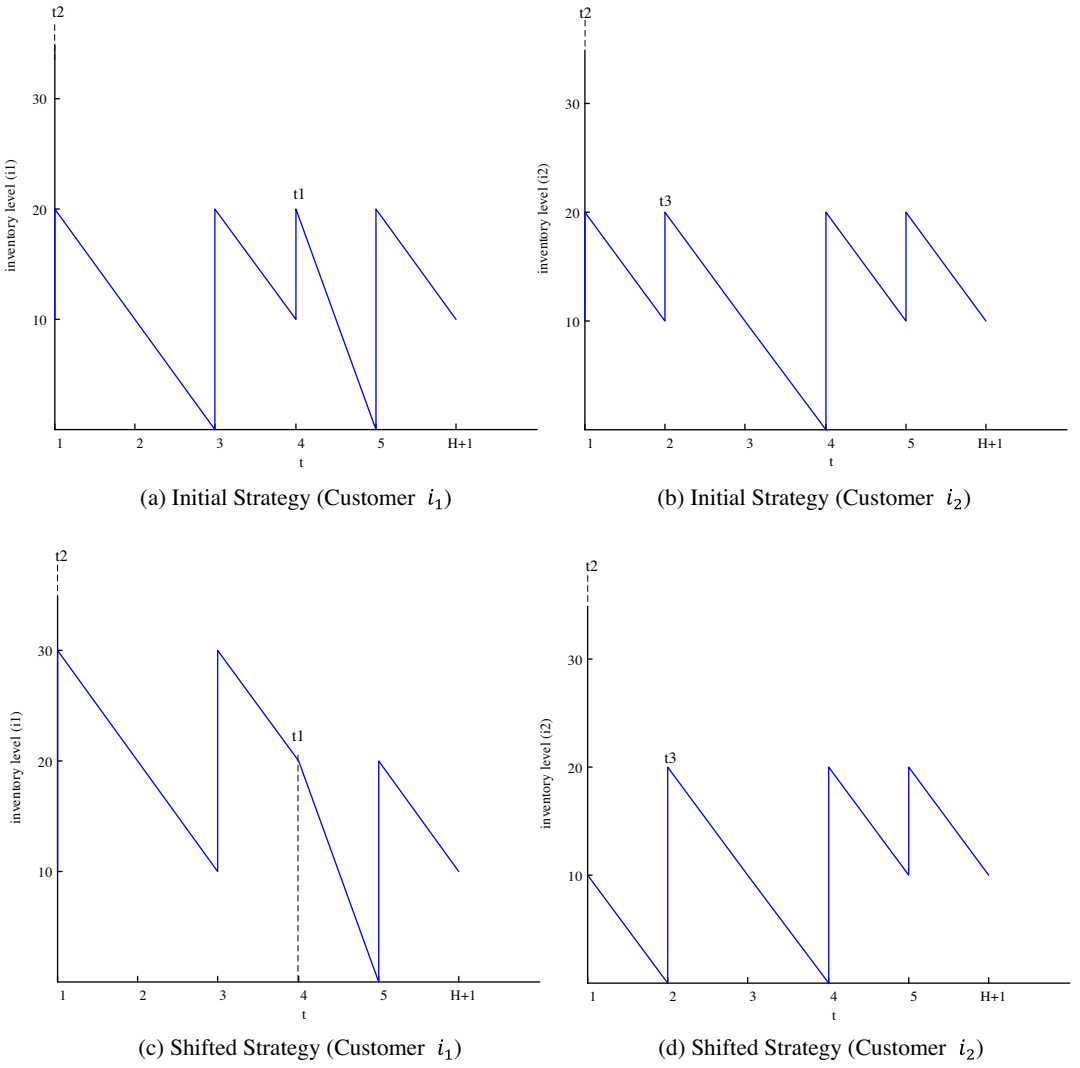
4.3.1.3. Shift-Delivery-Deepest operator. This operator is the extension of Shift-Delivery operator, in which we are interested in completely removing the delivery quantity of customer i on period t_1. First, we randomly choose a customer and the to-be shifted period t_1 such that $q_{t1} > 0$. Sort the other $H - 1$ periods in descending order of the delivery quantity, and gain a set H'. Finally, we shift the delivery on t_1 to t_2, to $t_3 \ldots$ until $q_{t1} = 0$ or each period is traversed. This operator performs significantly on decreasing the number of deliveries. This process can be defined as $Shiftest(i, t_1) = \bigcup_{t_2 \in H'} Shift(i, t_1, t_2)$.

Fig. 5 illustrates an example of this operator. Based on the replenishment plan shown is Fig. 1, we try to remove the delivery on $t_1 = 5$ with all the quantity of $q_{t1} = 20$. Constrained by the capacity available in different routes, this operator



(a) $t_2 = 2, q_{t2} = 0$ (b) $t_2 = 1, q_{t2} > 0$

Fig. 3. Examples of Shift-Delivery operator.



(a) Initial Strategy (Customer i_1)

(b) Initial Strategy (Customer i_2)

(c) Shifted Strategy (Customer i_1)

(d) Shifted Strategy (Customer i_2)

Fig. 4. Two-Shift-Delivery: $t_1 = 4$, $t_2 = 1$, $t_3 = 2$.

moves 10 of q_{t1} to $t_2 = 1$ and $t_3 = 4$ respectively. Representing in a more intuitive way, the "delivery array" changes from [100201020] to [20020200].

This operator is able to shift the delivery quantity from one period to multiple periods, trying to remove it as much as possible, while it operates only on a single period by Shift-Delivery operator.

4.3.1.4. Customer-Reassignment operator. Guerrero et al. (2013) proposed a depot reallocation neighborhood by shifting a customer to another depot, while keeping its inventory policies the same. However, by our test, we find it rarely works well. For example, if customer i is delivered on period t_1 with quantity q_{t1} by depot j_1 before the shift. We now try to reallocate this customer to depot j_2. According to the previous criteria, customer i should also be delivered on t_1 with the same quantity q_{t1}. But we cannot guarantee that $R(j_2, t_1)$ has enough capacity to accommodate this delivery. Actually this happens frequently, leading to an infeasible solution.

Here we make some improvements: the replenishment plan of the reallocated customers can be changed according to the minimum insertion cost criterion. The following model (**Reassign**) is to select the "best" route in $R(j_2, t)$ to serve customer i on each period $t \in H$ to minimize the total insertion costs, such that the capacity constraint of each inserted route and the total demand of customer i are both satisfied.

Suppose the number of routes of the new depot j_2 on period t is n_t. The total demand of customer i is D. For depot j_2, the maximum available capacity of route r on period t is Q_{tr}, while the minimum insertion cost in this route is c_{tr}. Binary decision variables $x_{tr} = 1$ mean that this customer is reallocated to route r on period t, the associated delivery quantity is q_{tr}. A mathematical model (**Reassign**) can be constructed as follows:

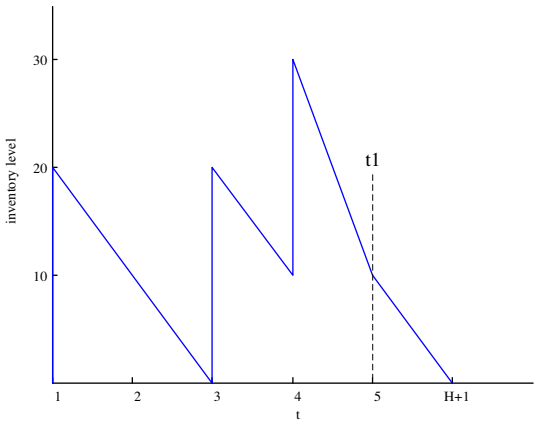


Fig. 5. Shift-Delivery-Deepest operator: $t_1 = 5$.

$$(\textbf{Reassign})\ \min = \sum_{t\in H}\sum_{r\in n_t} c_{tr}x_{tr} \tag{34}$$

Subject to

$$\sum_{r\in n_t} x_{tr} \leqslant 1 \quad \forall t \in H \tag{35}$$

$$q_{tr} \leqslant Q_{tr} \quad \forall t \in H,\ r \in n_t \tag{36}$$

$$\sum_{t\in H}\sum_{r\in n_t} q_{tr} = D \tag{37}$$

$$x_{tr} \leqslant q_{tr} \leqslant Dx_{tr} \quad \forall t \in H,\ r \in n_t \tag{38}$$

We are aimed at minimizing the total insertion cost, which is shown in the objective function (34). This customer can be served no more than once on each period is guaranteed by (35). (36) are the capacity constraints of each route. Constraints (37) ensure that the total demand of this customer should be satisfied. The set of Eq. (38) state that this customer has some delivery quantity in route r on period t only if it is allocated accordingly.

This formulation can be solved to optimality via GUROBI or CPLEX easily. Every time we want to reassign a customer to another depot, we solve this subproblem. The replenishment plans change with reallocation simultaneously, adding perturbation and mutation in the inventory-routing phase.

4.3.1.5. Customers-Exchange operator. Two customers allocated to different depots are exchanged. Guerrero et al. (2013) applied this operator remaining their inventory policies unchanged, while here we update their allocation and replenishment plans simultaneously. Two customers are chosen randomly and reallocated to each other's depots. Customer-Reassignment operator is used twice here. Note that each depot's capacity is not violated over the planning horizon.

4.3.2. Procedure of intensification

The intensification component is embedded in a simulated annealing (SA) framework, illustrated in Table 4. Parameter α is the cooling rate, while t_m and t_0 are initial and freezing temperatures, respectively. l_m denotes the number of iterations the search proceeds at a particular temperature.

Let RN_i $(i = 1, 2, \ldots, 5)$ denote the location-inventory-routing neighborhoods defined by the five operators introduced in Section 4.3.1. In the inner cycle of the SA algorithm, at each iteration, a neighborhood structure RN is randomly selected, and the whole neighborhood solutions using this structure are all explored. Then the best non-tabu move is employed. We refer this strategy as *best neighborhood*, comparing to the strategy used in basic SA (refer as *random neighborhood*, which randomly select one neighborhood). To "remember" the accepted move, a tabu list is used to avoid reverse moves. In lines 16–24, we adopt an *elitism strategy* to improve convergence speed. If the current solution has not been improved for C_m iterations, the search continues with the best solution (line 23).

Whenever a move is accepted, a *Vehicle Routing* is employed to improve the routing schedule of the affected periods. Note that in line 10, not every period whose inventory policy has changed need a *Vehicle Routing*. For example, Shift-Delivery operator will affect the inventory policies on t_1 and t_2 of a given depot j. If the selected customer i is already served on period t_2 $(q_{it2} > 0)$ before the operation, then the shifted quantity from t_1 to t_2 will be added to t_2. In such circumstance, routes $R(j, t_2)$ keep the same. Likewise, this situation happens in Two-Shift-Delivery operator and Shift-Delivery-Deepest operator. Or the following theorem holds:

Table 4
Pseudocode of intensification procedure.

Procedure: intensification (S_0)

1. Initialize t_m, t_0, l_m, C_m, α
2. $t = t_m$, $S^* = S_0$
3. **while** ($t > t_0$)
4. Set the cost of current best solution $c_0 = Cost(S^*)$
5. **for** $l = 0$: l_m
6. Randomly select RN from $\{RN_1, RN_2, RN_3, RN_4, RN_5\}$
7. Get a neighborhood solution by RN, identify the change of cost Δc
8. if ($hc < 0$ or $exp(-\frac{c}{t}) > random(0,1)$)
9. Implement the change: set $S_0 \leftarrow \boldsymbol{RN}(S_0)$
10. Re-optimize the affected routes: $S_0 \leftarrow$ vehicle routing (S_0)
11. end if
12. **if**(Cost(S_0) < $Cost(S^*)$)
13. Update the best solution: $S^* = S_0$
14. end if
15. end for
16. **if**(abs(Cost(S^*) − c_0) < 0.01)
17. counter = counter + 1
18. else
19. counter = 0
20. end if
21. **if**(counter > C_m)
22. counter = 0
23. set current solution $S_0 = S^*$
24. end if
25. $t = t \times \alpha$
26. end while
27. return S^*

Theorem 2. *If a customer is already served on the targeted period t, then after the shift, the routes on period t are still optimal.*
Proof of this proposition is obvious, so we ignore the details.

4.4. Post-optimization

In this section, we design three diversity operator: *Random-Initialization*, *Depot-Exchange* and *Random-Reassignment* to add disturbance to location-allocation decisions.

4.4.1. Diversity operators

4.4.1.1. Random-Initialization. Remember that in **(CFLP)** model presented in Section 4.1, the allocations change with the distance matrix $[c_{ij}]$. Inspired by the learning process on the choice of depots introduced in Prins et al. (2006), we propose the following method, named as *SetDistance*, to change the value of matrix $C' = [c'_{ij}]$ with an adaptive learning mechanism:

$$c'_{ij} = c_{ij} \cdot \frac{r + s_{ij}}{s_{ij}} \quad \forall i \in I, \forall j \in J \tag{39}$$

c_{ij} is the real distance between customer i and depot j. r is a uniform random number between [0, 1], while each s_{ij} is initialized to 1. Once a loop is traversed within the three steps (*Random-Initialization*, *Depot-Exchange* and *Random-Reassignment*), matrix $[s_{ij}]$ is updated: if customer i is assigned to depot j, $s_{ij} = s_{ij} + 1$, otherwise remain unchanged.

4.4.1.2. Depot-Exchange. One of the opened depot j is selected by the roulette-wheel method (sorted by the ratio: capacity/(used capacity)) and set to be closed, and all of its customers are reallocated among the remaining opened depots. If the remaining capacities of the opened depots are not enough for serving the customers of j, then we randomly select a currently closed depot. The Customer-Reassignment operator introduced in Section 4.3.1 is used here. If none previously closed depot is required to open, then the fixed cost of j will be subtracted from the total cost.

4.4.1.3. Random-Reassignment. Randomly select a set of customers with a fixed percentage γ given in advance. Then they are sorted in descending order by their total demands. Reassign these customers to a different open depot using the Customer-Reassignment operator one by one. If reallocation is not feasible for the set of opened depots, a new random depot is opened.

4.4.2. Procedure of post-optimization

Given the best solution S_0 found by the intensification component, a dedicated post-optimization procedure illustrated in Table 5 to diversify location-allocation is given. Post-optimization is realized by executing several loops. Each loop contains

Table 5
Pseudocode of post-optimization procedure.

Procedure: post-optimization (S_0)
1. Set re-optimization loops n_m
2. $S^* = S_0$
3. **for** $i = 0$: $3 \times n_m$
4. $a = i\%3$
5. **if**($a = 0$)
6. $C' \leftarrow$ SetDistance (S_0)
7. Re-initial $S_0 \leftarrow$ initialization (C')
8. else if($a = 1$)
9. $S_0 \leftarrow$ Depot-Exchange (S_0)
10. else if($a = 2$)
11. $S_0 \leftarrow$ Random-Reassignment (S_0)
12. end if
13. $S_0 \leftarrow$ intensification (S_0)
14. **if**(Cost(S_0) < *Cost*(S^*))
15. $S^* = S_0$
16. end if
17. end for
18. return S^*

these three diversity operators in sequence, each operator is followed by an intensification procedure. So each loop has three intensification procedures.

5. Computational results

The proposed SA-Hyb-ILRP is implemented in C# language and run on a PC with an AMD Sempron(tm) Processor (2.30 GHz) and 2 GB memory. The linear models ((**Reassign**) and the Lagrangian subproblems of (**CFLP**)) are solved by GUROBI 5.6. In this section, computational tests are first carried out on two series of benchmark problems (LRP and IRP). And then some ILRP instances are randomly generated. A comparison among SA-Hyb-ILRP, GUROBI and a standard sequential approach will be evaluated on computation effectiveness and efficiency. For each instance, the heuristic is run 5 times, and then the average objective value is reported in the tables.

5.1. Applying the heuristic to an LRP benchmark

This section evaluates the performance of proposed SA-Hyb-ILRP on capacitated LRP. LRP proposes to optimize location decisions simultaneously with routing decisions, the formulation of which can be easily obtained from (**ILRP**) by setting the planning horizon $H = 1$ and inventory holding cost $h_i = 0$ for each customer $i \in I$. Decision variables q and w are also excluded.

Some computational results are analyzed on classical instances that most researchers have used (see http://sweet.ua.pt/sbarreto/private/SergioBarretoHomePage.htm). The number of candidate depot sites ranges from 2 to 15 while the numbers of customers range from 12 to 318. All depots and routes are capacitated.

To consider efficiency of the presented heuristic method, the results are compared to several published results in Table 6.

In Table 6, first column indicates the author of the instance, the number of customers and the number of potential depot sites. The second column shows the capacity of the used vehicle. The best known solutions (BKS) are statistically reported in the literature. The solutions obtained by the clustering based heuristic (CH) in Barreto et al. (2007), the variable neighborhood descent based heuristic (VND) in Jabal-Ameli et al. (2011), the simulated annealing heuristic (SA) in Yu et al. (2010), and the multiple ant colony optimization algorithm (MACO) in Ting and Chen (2013) are shown in columns 4–9. Other methods also tested this dataset, including LRGTS, GRASP, MA|PM, etc., but they only computed 13 instances. We ignore them as direct comparisons may be in many cases biased. The last three columns shows the solutions obtained from the proposed SA-Hyb-ILRP. The "Gap" column shows the quality of the obtained solution as percent of deviation from BKS. The computational time needed (in seconds) for finding the best solution by corresponding methods are listed in the "CPU(s)" column.

As can be seen from Table 6, the proposed algorithm yields 12 best known solutions out of 19 instances. Our solutions are 2.56% above the BKS on average while the largest gap is 3.11%. Considering solution quality, SA-Hyb-ILRP outperforms CH and VND; while compared with the state of the art algorithms SA and MACO, SA-Hyb-ILRP is somewhat inferior. But the computational time is considerable. SA needs an average time of 464.1 s, while it is 215.6 s with SA-Hyb-ILRP. Besides, when solving those 19 instances, the maximal computing time are 3352.5 s and 1986.3 s, respectively, while it is only 1465.3 s with our heuristic.

The computational speed may depend on various factors, such as the CPU of the machines, the operation system, the compiler, the computer program, and the precision used during the execution of the run. Therefore, it is not an easy task to establish a fair comparison of the efficiency of various algorithms (Yu et al., 2010). As a matter of fact, our heuristic is

Table 6
Computational results for standard LRP instances.

Instance	Veh. Cap.	BKS	CH	VND[a]	SA[b]		MACO[c]		SA-Hyb-ILRP[d]		
					Cost	CPU(s)	Cost	CPU(s)	Cost	Gap (%)	CPU(s)
Christofides69-50×5	160	565.6	582.7	601	**565.6**	52.8	**565.6**	29.16	**565.62**	0.00	16.37
Christofides69-75×10	140	844.4	886.3	898	848	126.8	844.88	58.72	847.99	0.43	62.12
Christofides69-100×10	200	833.4	889.4	918	838.3	330.8	836.75	83.92	848.84	1.85	78.29
Daskin95-88×8	9000,000	355.8	384.9	383	**355.8**	226.9	**355.8**	99.53	**355.78**	−0.01	104.52
Daskin95-150×10	8000,000	43,919.9	46,642.7	47,662	45,109.4	577	44,131.02	166.95	44,397.22	1.09	193.1
Gaskell67-21×5	6000	424.9	435.9	451	**424.9**	18.3	**424.9**	5.64	**424.9**	0.00	23.74
Gaskell67-22×5	4500	585.1	591.5	591	**585.1**	16.6	**585.1**	4.69	**585.11**	0.00	21.57
Gaskell67-29×5	4500	512.1	**512.1**	**512**	**512.1**	23.9	**512.1**	8.59	**512.1**	0.00	32.9
Gaskell67-32×5	8000	562.2	571.7	583	**562.2**	27	**562.22**	13.05	**562.22**	0.00	45.8
Gaskell67-32×5	11,000	504.3	511.4	522	**504.3**	25.1	**504.3**	9.56	**504.33**	0.01	88.41
Gaskell67-36×5	250	460.4	470.7	492	**460.4**	31.7	**460.4**	13.14	**460.37**	−0.01	73.14
Min92-27×5	2500	3062	**3062**	3068	**3062**	23.3	**3062**	8.59	**3062.02**	0.00	10.54
Min92-134×8	850	5709	6238	6292	**5709**	522.4	**5709**	136.63	5743.49	0.60	196.87
Perl83-12×2	140	204	**204**	**204**	**204**	6.8	**204**	2.09	**203.98**	−0.01	6.45
Perl83-55×15	120	1112.1	1136.2	1144	1112.8	112.4	1112.58	28.81	**1112.06**	0.00	48.57
Perl83-85×7	160	1622.5	1656.9	1665	**1622.5**	213.1	1623.14	77.86	**1622.5**	0.00	189.54
Perl83-318×4	25,000	55,7275.2	580,680.2	602,315	563,493.1	2806.5	560,210.8	831.34	574,600.18	3.11	1465.3
Perl83-318×4	8000	67,0118.5	747,619	738,915	684,163.5	3352.5	**670,118.5**	1986.3	685,523.22	2.30	1086.7
Or76-117×14	150	12,290.3	12,474.2	13,451	12,434.5	323.4	12,355.91	77.45	12,392.26	0.83	352.31
Average		68,471.7	73,976.3	74,771.9	69,608.8	464.1	68,641	191.7	70,227.59	2.56	215.6

BKS: best solutions obtained in the literature.
Bold numbers indicate that best known solution values are attained by the corresponding approach.
[a] Coded in MATLAB R2007b, run on a PC with Intel Core 2 Duo CPU (2.0 GHz) and 2 GB memory.
[b] Coded in C language and run on a PC with Intel Core 2 Quad CPU (2.6 GHz) and 2 GB memory.
[c] Coded in C++, run on a PC with an Athlon XP 2500 + CPU (1.83 GHz) and 512 MB memory.
[d] Coded in C#, run on a PC with AMD Sempron(tm) CPU (2.30 GHz) and 2 GB memory.

mainly designed to solve the ILRP instances, or in other words, its "unprofessional" performance in solving LRP instances is understandable. Some time consuming calculation on inventory is embedded in the whole framework. Nevertheless, the computing efficiency and solution quality need to be improved, which is our next research point.

5.2. Applying the heuristic on two IRP benchmarks

IRP integrates the distribution process and the inventory control in a supply chain system, the model of which can be modified from (**ILRP**) by setting $J = \{0\}$ and all decisions $y_0 = 1, z_{i0} = 1$ for all $i \in I$. The efficiency of SA-Hyb-ILRP on IRP is evaluated by using the instances given in Zachariadis et al. (2009) and Qin et al. (2014). Each customer faces constant rate and variable rate of demand in two cases, respectively. Since there is only one depot, the Customer-Reassignment operator and Customers-Exchange operator embedded in the intensification procedure along with Post-optimization procedure are not required.

The second and third columns report the solutions computed by Zachariadis et al. (2009). Results of our heuristic for cost, total computation time and gap are presented in columns 4–6. Comparisons for the instances in Qin et al. (2014) are listed on the right of this table, with the same layout.

As shown in Table 7, SA-Hyb-ILRP practices a little poor solutions than those of Zachariadis et al. (2009). The gap is 3.09% on average. It should, however, be noted that our heuristic can also tackle with the variable demand rate, which was not handled in Zachariadis et al. (2009). While compared with Qin et al. (2014), our heuristic produces all the better solutions, with a little more computation time.

5.3. Comparison and analysis on random ILRP instances

5.3.1. Instances and parameters

Since there are no available benchmark instances for the problem under consideration, 20 ILRP instances are randomly generated. They have the following features: J: 2 to 25 depots; I: 5 to 300 customers; H: {3, 5, 7} periods. These instances are named as J–I–H, where J is the number of depots, I is the number of customers and H is the number of periods, as introduced in Section 2.

The generated instances are based on Guerrero et al. (2013), with a few modifications. Node locations are represented by coordinate pairs, which are uniformly distributed in [0, 100). Demand at customer i for period t is generated with a Normal distribution: $d_{it} \sim N(\mu_i, \sigma_j)$, where μ_i and σ_j are uniformly distributed in [5, 15] and [0, 5], respectively. Inventory holding costs for a single period at customers are randomly generated in the interval [0.05, 0.50]. Suppose that the total demand of all the customers over the planning horizon is D, and the maximal demand on a single period is $maxD$. Or more intuitively, $D = \sum_{i \in I} \sum_{t \in H} d_{it}$ and $maxD = \max_{i \in I, t \in H} d_{it}$. The opening cost for depot $j \in J$ are randomly generated with a Normal distribution with parameters (μ_i, σ_j) chosen from the set of pairs {(1000, 20), (5000, 100), (8000, 300), (10,000, 500)}. Depot capacity C_j is randomly generated in the interval $[D/10, D]$, guaranteeing that $\sum_{j \in J} C_j > D$. Vehicle capacity takes the uniform random values in the interval $[maxD, D/(2 \cdot H)]$.

A few parameters should be tuned. In the *Vehicle Routing* phase, parameters to be assigned have already been introduced in Section 4.2. In intensification procedure, parameters in the SA framework are cooling speed α , initial temperature t_m, freezing temperature t_0, consecutive non-improve iterations C_m and the iterations l_m at each temperature. While in the post-optimization procedure, re-optimization loops n_m is the only one. We perform a set of preliminary experiments in order to find appropriate parameter settings that produce overall good results across most instances, even if they are not the optimal settings for each instance. We consider a set of parameters and modify one, while keep the others fixed. The parameters tested include: $\alpha \in \{0.8, 0.85, 0.9, 0.95\}$, t_m:30 − 60 (in increments of 5 units), $C_m \in \{2, 3, 4, 5\}$, $l_m \in \{300, 400, 500, 600\}$ and the freezing temperature t_0 is fixed to 1. Among all the preliminary experiments we find the following parameter setting, illustrated in Table 8, provide the best results:

5.3.2. Computational results

For smaller instances, the comparison between a commercial solver GUROBI with a time limit of 2.5 h, the proposed SA-Hyb-ILRP and a sequential heuristic are shown in Table 9. The sequential heuristic is equivalent to a location-allocation (LAP) first, inventory–routing (IRP) second approach. In detail, the location-allocation decisions are first fixed by the **(CFLP)** model, and then the inventory-routing decisions are optimized through the intensification procedure described in Section 4.3.2. Note that Customer-Reassignment operator and Customers-Exchange operator are not used since they will change the location-allocation decisions.

In this table, the first column is the name of ten random generated instances. Columns 5, 7 and 10 present the cost of the best solution found within the time limit or given parameter settings using the corresponding methods. Columns 2–4 show the gap of lower and upper bound when solved by GUROBI within 100 s, 1200 s and 9000 s respectively, while the gap between SA-Hyb-ILRP and GUROBI, and between the sequential heuristic and GUROBI, are illustrated in columns 8 and 11. Columns 6, 9 and 12 are the computational times elapsed before the best solution is encountered.

On average, the proposed heuristic outperforms GUROBI by 4.14%, with an average running time of 316 s (Note that GUROBI uses 2.5 h). 5 out of 10 new best solutions are found and other five solutions have a gap less than 0.35%. If the same

Table 7
Computational results of SA-Hyb-ILRP on standard IRP instances.

Instance	Const demand rate					Variable demand rate				
	Zachariadis et al. (2009)		SA-Hyb-ILRP			Qin et al. (2014)		SA-Hyb-ILRP		
	BKS	CPU(s)	Cost	CPU(s)	Gap (%)	BKS	CPU(s)	Cost	CPU(s)	Gap (%)
p_30_7	33,004.4	170.8	33,594.8	76.48	1.79	34,764.3	196	**33,898.25**	87.96	−2.49
p_50_7	52,061.5	234.3	52,889.69	219.64	1.59	54,255.8	271	**53,453.86**	238.3	−1.48
p_70_7	53,478.7	453.9	55,457.42	316.29	3.70	60,822.6	466	**56,663.09**	288.32	−6.84
p_90_7	71,678.1	319.1	72,209.69	583.67	0.74	77,659.9	689	**73,281.31**	785.62	−5.64
p_110_7	73,858.8	934.4	76,539.36	897.65	3.63	87,951.4	738	**79,786.73**	1025.14	−9.28
p_130_7	85,213.1	1041.6	86,632.56	1258.4	1.67	98,777.8	1051	**89,527.54**	1587.6	−9.36
p_150_7	116,306.5	1347.7	121,365.71	1508.32	4.35	136,673	1517	**12,3493.88**	1897.25	−9.64
p_170_7	135,845.4	1563.7	143,102.67	1857.63	5.34	168,469	1921	**15,7274.93**	1965.74	−6.64
p_190_7	168,548.9	1955.4	176,412.68	2047.21	4.67	205,384	2375	**18,4316.3**	2245.87	−10.26
p_210_7	187,869.1	2321.5	194,341.83	2657.6	3.45	229,354	2221	**206,507.38**	2748.21	−9.96
Average	97,786.4	1034.24	101,254.641	1142.289	3.09	115,411.2	1144.5	**105,820.33**	1287.001	−7.16

Bold numbers indicate that better solutions are attained by the corresponding approach.

Table 8
Parameter settings for SA-Hyb-ILRP.

Parameters	α	t_m	t_0	C_m	l_m	n_m
Values	0.85	40	1	4	400	7

We then test our SA-Hyb-ILRP with these identical parameter values.
Bold numbers indicate that better solutions are attained by the corresponding approach.

Table 9
Comparisons for random ILRP instances.

Instance	GUROBI					SA-Hyb-ILRP			Sequential heuristic		
	GAP (100 s)	GAP (1200 s)	GAP (9000 s)	COST	CPU(s)	COST	GAP (%)	CPU (s)	COST	GAP (%)	CPU (s)
2-5-3	0.86	0	0	9958.71	130	**9958.98**	0.00	15.96	10,363.16	4.06	1.14
3-8-3	6.79	6.28	3.65	6771.4	80	6774.87	0.05	46.8	6799.58	0.42	5.1
3-8-5	54.9	5.39	4.77	17,652.46	7090	17,654.66	0.01	74.21	19,485.83	10.39	6.52
3-8-7	47	6.08	5.53	14,247.75	8325	14,252.47	0.03	125.14	14,372.96	0.88	5.25
3-10-3	4.01	3.71	3.37	6525.18	8245	6530.9	0.09	260.32	7101.85	8.84	12.32
3-15-7	25.8	10.4	8.98	15,226.38	8940	**15,220.19**	−0.04	485.68	17,980.15	18.09	18.65
5-10-5	–	42	21.5	23,252.89	6650	**19,936.59**	−14.26	587.63	**20,070.7**	−13.69	20.12
5-10-7	27.7	23.1	22.4	3284.84	2700	3296.23	0.35	495.6	3709.88	12.94	14.23
5-15-5	88.2	15.9	12.3	3164.51	8915	**3143.41**	−0.67	523.65	4157.6	31.38	17.86
5-15-7	46.8	36.4	15.3	20,194.5	7980	**18,531.83**	−8.23	547.21	**18,820.99**	−6.80	19.85
Average	33.56	16.58	10.87	12,027.86	5905.50	**11,530.01**	−4.14	316.22	12,286.27	2.15	12.10

computing time allowed, we have reasons to believe that SA-Hyb-ILRP outperforms the solver on all instances. The sequential heuristic provides solutions that are about 2.15% worse than the ones found by GUROBI. Although the computing time of sequential heuristic is much smaller, we prefer to SA-Hyb-ILRP for it provides much better results.

For larger instances, Table 10 outlines the comparison between SA-Hyb-ILRP, the sequential heuristic and GUROBI with a time limit of 5 h. The GUROBI solver is able to find feasible solutions for only three out of 10 instances, leading to out-of-memory error for those seven larger instances. By relaxing the subtour elimination constraints, and adding cutting plane constrains in Eq. (40) which ensure that any customer node is not connected to itself in each route, instances 10-60-3, 8-70-5 and 8-80-5 are able to find a lower bound, while the other four larger instances still failed.

$$x_{iikt} = 0 \quad \forall i \in I, \forall k \in K, \forall t \in H \tag{40}$$

SA-Hyb-ILRP solves the instances with an average computational time of less than half an hour. The classical sequential heuristic is much faster, with an average time of 51 s, but the solutions are 17% worse than SA-Hyb-ILRP. We also notice that SA-Hyb-ILRP can effectively reduce the number of opened depots needed, which can be ascribed to the post-optimization procedure.

Compared to the exact solver, the proposed SA-Hyb-ILRP can reach a considerable solution for large-scale problems, while the former cannot work out. Take instance 10-60-3 as an example, we can see that that gap between SA-Hyb-ILRP and best solution is 22% at most, which is acceptable considering the running time.

5.3.3. Convergence analysis

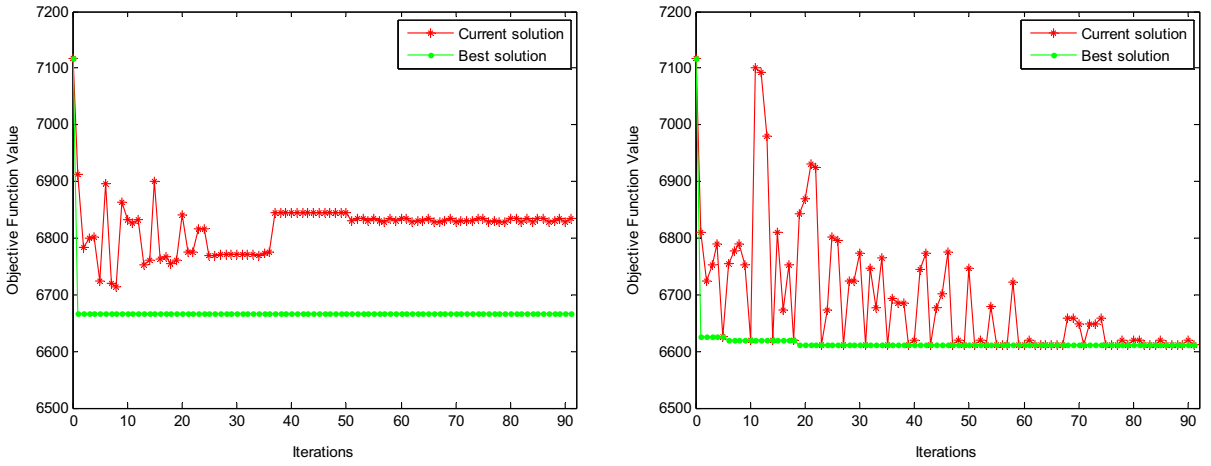
Remember in Section 4.3.2 we proposed a *best neighborhood* strategy to select a neighborhood solution, and *elitism strategy* to escape from local optimum. In this section, we analyze how these strategies improve the convergence speed of SA-Hyb-ILRP.

We first use *random neighborhood* and try two cases on instance 3-10-3: with or without *elitism strategy*. Cooling speed α is reset as 0.96, the initial temperature $t_m = 40$ and freezing temperature $t_0 = 1$, so a maximal iterations $\lceil \log_\alpha \frac{t_0}{t_m} \rceil = 91$ would be iterated. As Fig. 6 shows, without the *elitism strategy*, the algorithm produces a solution of 6663.35 at the initial temper-

Table 10
Performance on larger ILRP instances.

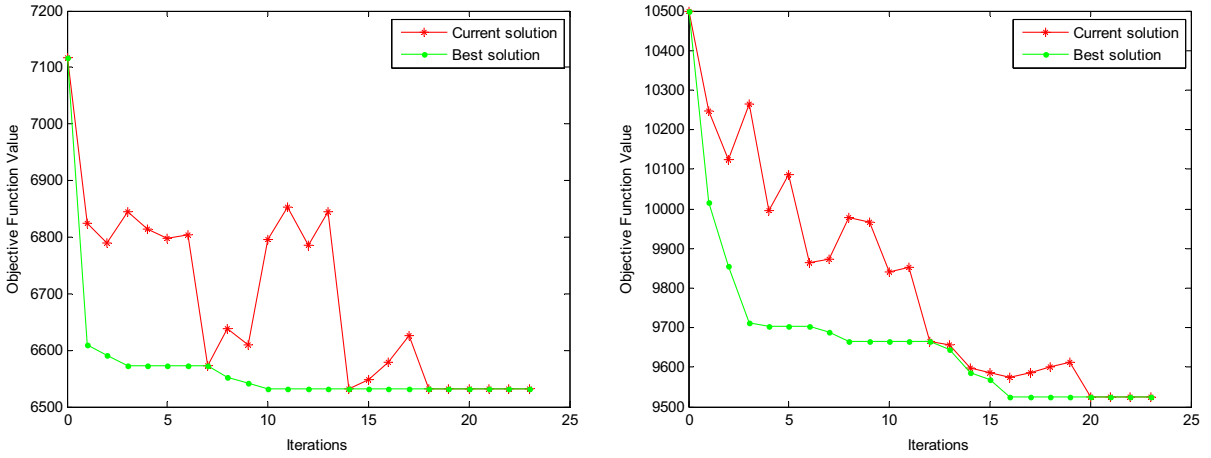
Instance	GUROBI		SA-Hyb-ILRP				Sequential heuristic			
	Cost	CPU(s)	Cost	[b]Depot	GAP (%)	CPU(s)	Cost	[b]Depot	GAP (%)	CPU(s)
5-30-3	9822.7	7145	**8343.27**	2	−15.06	587.6	**8402.36**	2	−14.46	18.52
5-40-5	29,382.12	13,733	**13,507.89**	2	−54.03	698.52	**13,919.62**	2	−52.63	26.5
8-50-7	21,742.89	13,589	**10,127.58**	2	−53.42	714.3	**19,341.65**	3	−11.04	28.65
10-60-3[a]	3135.83	4652	3837.94	2	22.39	695.25	4148.37	3	32.29	20.2
8-70-5[a]	10,705.59	1388	12,391.97	2	15.75	498.63	12,794.09	2	19.51	16.5
8-80-5[a]	8260.82	6941	9520.99	3	15.25	785.6	11,030.4	4	33.53	29.85
15-100-7	–	–	27,761.56	6	–	1236.21	37,728.64	7	–	30.98
5-120-5	–	–	28,938.4	4	–	1042.68	37,906.5	5	–	80.35
20-150-7	–	–	46,148.96	7	–	1562.3	55,912.54	9	–	58.47
25-300-7	–	–	87,186.54	14	–	2365.87	88,003.22	15	–	205.85
Average	13,841.66	7908.00	24,776.51	4.40	−11.52	1018.70	28,918.74	5.20	1.20	51.59

Bold numbers indicate that better solutions are attained by the corresponding approach.
[a] Subtour elimination constraints relaxed.
[b] The number of opened depots in the solution.



(a) Without *elitism strategy* (35.4s) (b) With *elitism strategy* (52.36s)

Fig. 6. Effect of elitism strategy (*random neighborhood*, 3-10-3).



(a) 3-10-3 (45.32s) (b) 8-80-5 (142.57s)

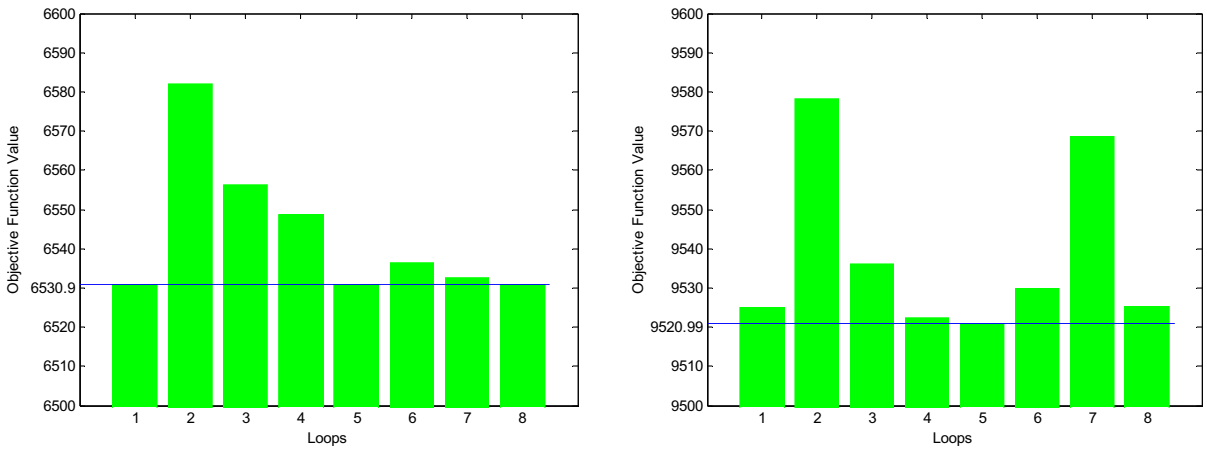
Fig. 7. Convergence curves of intensification procedure (*best neighborhoods*, with *elitism strategy*).

ature 40 (in its inner cycle). However in the subsequent search process, the best solution obtains no further improvement. The reason maybe that as the temperature cools down, the probability of accepting deteriorations is decreased. While with *elitism strategy*, at the 18th iteration (say, the temperature is $40 \times 0.96^{17} = 19.98$), since the best solution has not been improved for a consecutive $C_m = 4$ iterations, the current solution is reset as the best solution. Consequently, a better solution is explored at the 19th iteration. So the elitism strategy may let the current solution "stay focused" and not drift too far.

We then use both *best neighborhood* strategy and *elitism strategy* on a small-sized instance "3-10-3" and a middle-sized instance "8-80-5", to evaluate the convergence of our algorithm. A single intensification procedure and a post-optimization procedure are tested and illustrated in Figs. 7 and 8, respectively.

In detail, whenever an inner cycle ends at each temperature (just after line 24 in Table 4), the objective function value of current solution and best solution are recorded. As the temperature cools, a total number of $\lceil \log_{0.85} \frac{1}{40} \rceil = 23$ iterations are executed. Fig. 7 shows the curve of objective function values. We see that the algorithm is very efficient and converges very fast. Also, due to the using of *elitism strategy*, at the 7th iteration of Fig. 7a and the 12th iteration of Fig. 7b, a much better solution can be reached.

Note that in the titles of Figs. 6 and 7, the total computation time to run a single intensification procedure (SA framework) are marked. For instance 3-10-3, Fig. 6b and Fig. 7a both use the *elitism strategy*, but the computation time of Fig. 7a (45.32 s, *best neighborhood*) is lower than that of Fig. 6b (52.36 s, *random neighborhood*). This is because with *random neighborhood*, in one iteration, a not good enough neighborhood may be accepted and an "unnecessary" *Vehicle Routing* procedure may be



(a) 3-10-3 (b) 8-80-5

Fig. 8. Effect of post-optimization procedure (*best neighborhood*).

employed. In most cases, the resulted solution is abandoned and too many iterations must be explored to ensure the convergence. While with *best neighborhood*, only those good enough neighborhood solutions are accepted, and so the routing process is more "valuable".

Proposed post-optimization procedure has 7($n_m = 7$) loops. We record the solution when the first intensification procedure ends (just after line 2 in Table 2), as well as those seven solutions each time when one loop ends in the post-optimization procedure (inside line 3 in Table 2). To test the effect of post-optimization, for instance 3-10-3 and 8-80-5, the bar charts each with 8 values are shown in Fig. 8a and b, respectively. For instance 3-10-3, we see that the best solution has been reached twice even when big disturbance are imposed, showing that the intensification procedure is effective; while for 8-80-5, the first solution is updated twice during post-optimization phase, proving the effectiveness of our post-optimization approach.

From these comparisons, we see that the improvement to the basic SA algorithm: *elitism strategy*, *best neighborhood* strategy, and post-optimization, are very helpful for the convergence of the algorithm, for example, we can often get a good enough solution in the first five iterations.

6. Conclusions

The main contribution of this paper is to simultaneously optimize location, allocation, inventory and routing decisions. Our paper differs from the earlier literature on ILRP in these aspects: (1) potential depot sites and vehicle fleets are capacitated; (2) replenishment policies and routing arrangement are determined by the supplier under the vendor managed inventory mode; (3) multi-period case: each customer faces non-constant and periodic demand over a given planning horizon; (4) inventory and routing are optimized in an integrated manner, not limited to several specific inventory policies.

In particular, this paper designs a hybrid metaheuristic, named SA-Hyb-ILRP, to solve this integrated supply chain design problem. Even for large instances, it is proved to be efficient and effective. The simplified method to evaluate inventory cost, the well-designed neighborhood operators, as well as the way of subproblem decomposition, can explain to the advantages we achieve. The proposed algorithm can also be adopted to solve the LRP and IRP with little modifications.

For future research, a few improvements can be attempted to obtain better results and lower running time. Various spatial neighbor-reduction strategies such as kth nearest neighbor search, Voronoi neighborhoods, and k-ring-shaped sets of Voronoi neighbors can be used to reduce the search effort required for vehicle routing. More combination of parameters setting can be tuned. Moreover, since this is a multiple depot case, allowing vehicles to perform multi-trips may be an alternative to reduce fleet size and also routing cost. This paper is the first step towards a stochastic version supply chain design. Uncertainty may be also considered in the future research. Emergency situations such as rapid demand fluctuation and failure in depots broadly exist in logistics activities. To design a robust and reliable network will be a very interesting and challenging problem.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (71272030), as well as Science & Technology Foundation of Shenzhen City (CXZZ20130321145336439). The authors would like to thank the anonymous reviewers for their helpful and constructive comments that greatly contributed to improving the final version of the paper. They would also like to thank the Editors for their generous comments and support during the review process.

References

Ambrosino, D., Grazia Scutellà, M., 2005. Distribution network design: new problems and related models. Eur. J. Oper. Res. 165 (3), 610–624.
Anily, S., Federgruen, A., 1990. One warehouse multiple retailer systems with vehicle routing costs. Manage. Sci. 36 (1), 92–114.
Andersson, H., Hoff, A., Christiansen, M., Hasle, G., Lokketangen, A., 2010. Industrial aspects and literature survey: combined inventory management and routing. Comput. Oper. Res. 37 (9), 1515–1536.
Barreto, S., Ferreira, C., Paixao, J., Santos, B.S., 2007. Using clustering analysis location-routing in a capacitated problem. Eur. J. Oper. Res. 179 (3), 968–977.
Bertazzi, L., Paletta, G., Speranza, M.G., 2002. Deterministic order-up-to level policies in an inventory routing problem. Transp. Sci. 36 (1), 119–132.
Cha, B.C., Moon, I.K., Park, J.H., 2008. The joint replenishment and delivery scheduling of the one-warehouse, n-retailer system. Transp. Res. Part E Logist. Transp. Rev. 44 (5), 720–730.
Daskin, M.S., 1995. Network and Discrete Location: Models, Algorithms and Applications. Wiley-Interscience Publication, New York.
Guerrero, W.J., Prodhon, C., Velasco, N., Amaya, C.A., 2013. Hybrid heuristic for the inventory location-routing problem with deterministic demand. Int. J. Prod. Econ. 146 (1), 359–370.
Huang, S., Lin, P., 2010. A modified ant colony optimization algorithm for multi-item inventory routing problems with demand uncertainty. Transp. Res. Part E Logist. Transp. Rev. 46 (5), 598–611.
Jabal-Ameli, M.S., Aryanezhad, M.B., Ghaffari-Nasab, N., 2011. A variable neighborhood descent based heuristic to solve the capacitated location-routing problem. Int. J. Ind. Eng. Comput. 2 (1), 141–154.
Javid, A.A., Azad, N., 2010. Incorporating location, routing and inventory decisions in supply chain network design. Transp. Res. Part E Logist. Transp. Rev. 46 (5), 582–597.
Liu, S.C., Lee, S.B., 2003. A two-phase heuristic method for the multi-depot location routing problem taking inventory control decisions into consideration. Int. J. Adv. Manuf. Technol. 22 (11–12), 941–950.
Liu, S.C., Lin, C.C., 2005. A heuristic method for the combined location routing and inventory problem. Int. J. Adv. Manuf. Technol. 26 (4), 372–381.
Ma, H., Davidrajuh, R., 2005. An iterative approach for distribution chain design in agile virtual environment. Ind. Manage. Data Syst. 105 (6), 815–834.
Max Shen, Z.-J., Qi, L., 2007. Incorporating inventory and routing costs in strategic location models. Eur. J. Oper. Res. 179 (2), 372–389.
Mete, H.O., Zabinsky, Z.B., 2010. Stochastic optimization of medical supply location and distribution in disaster management. Int. J. Prod. Econ. 126 (1), 76–84.
Miranda, P.A., Garrido, R.A., 2008. Valid inequalities for Lagrangian relaxation in an inventory location problem with stochastic capacity. Transp. Res. Part E Logist. Transp. Rev. 44 (1), 47–65.
Moin, N.H., Salhi, S., 2006. Inventory routing problems: a logistical overview. J. Oper. Res. Soc. 58 (9), 1185–1194.
Nozick, L.K., Turnquist, M.A., 1998. Integrating inventory impacts into a fixed-charge model for locating distribution centers. Transp. Res. Part E Logist. Transp. Rev. 34 (3), 173–186.
Prins, C., Prodhon, C., Calvo, R.W., 2006. Solving the capacitated location-routing problem by a GRASP complemented by a learning process and a path relinking. 4OR 4 (3), 221–238.
Prodhon, C., Prins, C., 2014. A survey of recent research on location-routing problems. Eur. J. Oper. Res. 238 (1), 1–17.
Qi, M., Zhang, Y., Zhang, J., Miao, L., 2013. A new two-phase hybrid metaheuristic for vehicle routing problem with time windows. J. East Asia Soc. Transp. Stud. 10, 880–896.
Qin, L., Miao, L., Ruan, Q., Zhang, Y., 2014. A local search method for periodic inventory routing problem. Expert Syst. Appl. 41 (2), 765–778.
Rego, C., 1998. A subpath ejection method for the vehicle routing problem. Manage. Sci. 44 (10), 1447–1459.
Rennemo, S.J., Rø, K.F., Hvattum, L.M., Tirado, G., 2014. A three-stage stochastic facility routing model for disaster response planning. Transp. Res. Part E Logist. Transp. Rev. 62 (0), 116–135.
Reza Sajjadi, S., Hossein Cheraghi, S., 2011. Multi-products location-routing problem integrated with inventory under stochastic demand. Int. J. Ind. Syst. Eng. 7 (4), 454–476.
Solomon, M.M., 1987. Algorithms for the vehicle routing and scheduling problems with time window constraints. Oper. Res. 35 (2), 254–265.
Ting, C., Chen, C., 2013. A multiple ant colony optimization algorithm for the capacitated location routing problem. Int. J. Prod. Econ. 141 (1), 34–44.
Toth, P., Vigo, D., 2003. The granular tabu search and its application to the vehicle-routing problem. INFORMS J. Comput. 15 (4), 333–346.
Viswanathan, S., Mathur, K., 1997. Integrating routing and inventory decisions in one-warehouse multiretailer multiproduct distribution systems. Manage. Sci. 43 (3), 294–312.
Yu, V.F., Lin, S.W., Lee, W.Y., Ting, C.J., 2010. A simulated annealing heuristic for the capacitated location routing problem. Comput. Ind. Eng. 58 (2), 288–299.
Zachariadis, E.E., Tarantilis, C.D., Kiranoudis, C.T., 2009. An integrated local search method for inventory and routing decisions. Expert Syst. Appl. 36 (7), 10239–10248.