INFORMS Journal on Computing
Vol. 24, No. 1, Winter 2012, pp. 101–116
ISSN 1091-9856 (print) | ISSN 1526-5528 (online)

http://dx.doi.org/10.1287/ijoc.1100.0439

A Hybrid Heuristic for an Inventory Routing Problem

Claudia Archetti, Luca Bertazzi
Department of Quantitative Methods, University of Brescia, 25122 Brescia, Italy
{archetti@eco.unibs.it, bertazzi@eco.unibs.it}

Alain Hertz
Department of Mathematics and Industrial Engineering, École Polytechnique, CP 6079, Montréal, Québec H3C 3A7, Canada; and GERAD, École des HEC, Montréal, Québec H3T 2A7, Canada, alain.hertz@gerad.ca

M. Grazia Speranza
Department of Quantitative Methods, University of Brescia, 25122 Brescia, Italy,
speranza@eco.unibs.it

We consider an inventory routing problem in discrete time where a supplier has to serve a set of customers over a multiperiod horizon. A capacity constraint for the inventory is given for each customer, and the service cannot cause any stockout situation. Two different replenishment policies are considered: the order-up-to-level and the maximum-level policies. A single vehicle with a given capacity is available. The transportation cost is proportional to the distance traveled, whereas the inventory holding cost is proportional to the level of the inventory at the customers and at the supplier. The objective is the minimization of the sum of the inventory and transportation costs. We present a heuristic that combines a tabu search scheme with ad hoc designed mixed-integer programming models. The effectiveness of the heuristic is proved over a set of benchmark instances for which the optimal solution is known.

Key words: inventory routing problem; metaheuristic; tabu search; optimization; integer programming
History: Accepted by David Woodruff, Area Editor for Heuristic Search and Learning; received February 2009; revised July 2010; accepted September 2010. Published online in *Articles in Advance* February 2, 2011.

1. Introduction

The class of inventory routing problems (IRPs) includes a variety of different optimization problems that, although often very different from each other, all consider a routing and an inventory component of an optimization problem. Time may be discrete or continuous, demand may be deterministic or stochastic, and inventory holding costs may or may not be accounted for in the objective function. When the holding costs are not included in the objective function, usually a limited inventory capacity at the customers is available and cannot be exceeded. Inventory routing problems have received little attention compared with vehicle routing problems. However, the interest in this class of problems has been increasing since the early 1980s. Some pioneering papers appeared in the 1980s (see, e.g., Bell et al. 1983, Blumenfeld et al. 1985, Dror et al. 1985, Ferdergruen and Zipkin 1984, Golden et al. 1984), whereas several papers appeared in the last two decades, and some surveys (see, e.g., Bertazzi et al. 2008, Campbell et al. 1998, Cordeau et al. 2009, Federgruen and Simchi–Levi 1995) summarize the state of the art. The field is now mature enough to stimulate research efforts devoted to the design of effective algorithms for some known IRP. A branch-and-cut algorithm that makes use of some sets of valid inequalities is presented in Archetti et al. (2007) for an IRP. The availability of exact solutions for test instances makes possible the evaluation of the performance of a heuristic algorithm. In this paper, we present a heuristic for the solution of this IRP that combines a tabu search scheme with ad hoc designed mixed-integer programming (MIP) models. In fact, whereas tabu search algorithms have been proven to be very effective for many vehicle routing problems, the complexity of the IRP we studied required a more sophisticated heuristic search of the solution space to get high-quality solutions. The effectiveness of the heuristic is proved over a set of benchmark instances.

The rest of this paper is organized as follows. In §2 we describe the IRP we study. In §3 we describe the hybrid heuristic we propose, and in §4 we mention the computational experiments and show the obtained results.

2. Problem Description

We consider a distribution network where a product is shipped from a common supplier, denoted by 0, to a

set $\mathcal{M} = \{1, 2, \ldots, n\}$ of customers over a time horizon of H periods. At each discrete time $t \in \mathcal{T} = \{1, \ldots, H\}$ a quantity r_{0t} is produced at the supplier, and a quantity r_{it} is consumed at customer $i \in \mathcal{M}$. A starting inventory level B_0 at the supplier is given. Each customer i has a maximum capacity U_i and a given starting inventory $I_{i0} \leq U_i$. If customer i is visited at time t, then the quantity x_{it} shipped to the customer depends on the replenishment policy. We consider two different replenishment policies. In the order-up-to-level (OU) policy, if customer i is served at time t, the quantity x_{it} is the difference between U_i and the current inventory level I_{it} of i. In the maximum-level (ML) policy, the quantity x_{it} can take any nonnegative value that does not violate the capacity U_i. The OU policy is a well-known policy in inventory management. The ML policy is interesting only in integrated inventory routing systems and was introduced in Archetti et al. (2007). The name used indicates that only a constraint on the maximum inventory level has to be satisfied.

The inventory holding cost is charged both at the supplier and at the customers. Inventory holding costs at the customers are taken into account in real inventory routing problems whenever the supplier agrees with the customers to require the payment of the products only when the products are sold by the customers. This is one of the incentives used by suppliers to move toward inventory routing systems. Denoting by h_0 the unit inventory cost at the supplier and by B_t the inventory level at the supplier at time t, the total inventory cost at the supplier is $\sum_{t \in \mathcal{T}'} h_0 B_t$, where $\mathcal{T}' = \mathcal{T} \cup \{H+1\}$. The time $H+1$ is included in the computation of the inventory cost to take into account the consequences of the operations performed at time H. Denoting by h_i the unit inventory cost of customer $i \in \mathcal{M}$, the total inventory cost over the time horizon is $\sum_{t \in \mathcal{T}'} h_i I_{it}$. Shipments from the supplier to the customers can be performed at each time $t \in \mathcal{T}$ by a vehicle of capacity C. The transportation cost c_{ij} from i to j is known, and $c_{ij} = c_{ji}$, $i, j \in \mathcal{M}' = \mathcal{M} \cup \{0\}$. Therefore, letting y_{ij}^t be a binary variable equal to 1 if j immediately follows i in the route traveled at time t and 0 otherwise, the total transportation cost is $\sum_{i \in \mathcal{M}'} \sum_{j \in \mathcal{M}', i \neq j} \sum_{t \in \mathcal{T}} c_{ij} y_{ij}^t$. Note that once the customers to be replenished are determined, a traveling salesman problem (TSP) has to be solved in each period. The notation used is summarized in the online supplement (available at http://joc.journal.informs.org/).

To be *feasible*, a solution should not have any stockout at the supplier and at the customers (i.e., $B_t \geq 0$ and $I_{it} \geq 0$ for all $t \in \mathcal{T}'$ and $i \in \mathcal{M}$), the level of the inventory of each customer i should not be greater than its maximum level U_i, and the total quantity delivered at any given time should not exceed the vehicle capacity C. The objective of the considered IRP is to determine a feasible solution with minimum total cost. If the initial inventory of each customer is 0 and $H = 1$, the problem becomes a TSP and, thus, is NP-hard.

3. HAIR—A Hybrid Heuristic

Tabu search is a local search technique that visits a search space S by moving step by step from a current solution $s \in S$ to a *neighbor* solution $s' \in N(s)$, where $N(s)$ is a subset of S called the *neighborhood* of s. A *tabu list* forbids some moves that would bring the search back to a recently visited solution. Tabu search was introduced in Glover (1986). A description of the method and its concepts can be found in Glover and Laguna (1997).

To solve the IRP, we use a hybrid heuristic that combines a tabu search with the solution of MIP problems. We call the heuristic HAIR (hybrid approach to inventory routing). The general scheme of HAIR can be found in Algorithm 1.

Algorithm 1 (HAIR—A hybrid heuristic)

Apply the *Initialization* procedure to generate an initial solution s. Set $s_{\text{best}} \leftarrow s$.
while the number of iterations without improvement of $s_{\text{best}} \leq MaxIter$ **do**
  Apply the *Move* procedure to find the best solution s' in the neighborhood $N(s)$ of s.
  if s' is better than s_{best} **then**
    Apply the *Improvement* procedure to possibly improve s' and set $s_{\text{best}} \leftarrow s'$.
  end if
  Set $s \leftarrow s'$.
  if the number of iterations without improvement of s_{best} is a multiple of *JumpIter* **then**
    Apply the *Jump* procedure to modify the current solution s.
  end if
end while

Parameters *MaxIter* and *JumpIter* indicate when the algorithm should stop or jump to a new region of the search space. We use four basic procedures that will be described in details in the following sections: *Initialization* generates a starting solution in S; *Move* generates the best neighbor s' of s in $N(s)$; *Improvement* tries to improve s_{best} by solving MIP problems; and *Jump* performs changes on s in order to jump to a new region of the search space S. In the next subsections we describe the main ingredients of the proposed tabu search, the *Improvement* procedure based on the solution of two MIPs and the *Jump* procedure. We also give a proof that the considered MIPs are NP-hard problems.

3.1. Tabu Search Ingredients

As mentioned in §2, a solution to the IRP is feasible if there is no stockout, the level of the inventory of

each customer is not greater than its maximum level, and there is no violation of the vehicle capacity constraint. The search space S visited by the tabu search contains solutions that have no stockout at the customers, whereas stockout at the supplier is permitted, and the vehicle capacity is possibly exceeded at some time periods. Such solutions are said *admissible*. The objective function to be minimized is the sum of the inventory and transportation costs plus two penalty terms related to infeasibility. More precisely, given a solution $s \in S$, we denote the following:

- $B_t(s)$ the inventory level at the supplier at time t in s.
- $I_{it}(s)$ the inventory level of customer i at time t in s.
- $Q_t(s)$ the total quantity delivered at time t in s.
- $K_t(s)$ the transportation cost at time t in s.
- $T_i(s)$ the set of delivery times concerning customer i in s.

The value $f(s)$ of solution s is then defined as follows:

$$f(s) = \sum_{t\in\mathcal{T}'}\left(h_0 B_t(s) + \sum_{i\in\mathcal{M}} h_i I_{it}(s)\right) + \sum_{t\in\mathcal{T}} K_t(s) + \alpha\sum_{t\in\mathcal{T}}[Q_t(s) - C]^+ + \beta\sum_{t\in\mathcal{T}'}[-B_t(s)]^+,$$

where $[\cdot]^+ = \max\{\cdot, 0\}$, and α and β are penalty factors initialized to 1 and updated every 10 iterations as in Gendreau et al. (1994). If the last 10 solutions were all feasible with respect to the vehicle capacity (no stock-out at the supplier) constraint, then the value of α (β) is halved; if they were all infeasible, the value of α (β) is doubled. A solution is represented as a vector of routes of length H where each route is composed by the following components:

- A vector of visited vertices, listed in the order of visit (the first vertex is always the supplier), and
- a vector of quantities delivered to customers.

In the *Initialization* procedure, each customer from 1 to n is considered sequentially, and the delivery times are set as late as possible before a stockout situation occurs. Such a solution is obviously admissible but not necessarily feasible.

At each iteration, the algorithm moves from a solution $s \in S$ to a new solution s'. We consider two tabu lists, $\mathcal{L}_a$ and $\mathcal{L}_r$. If customer i is visited at time t in s but not in s', then the pair (i, t) is introduced in $\mathcal{L}_a$, and it is forbidden for some iterations to add delivery time t at customer i. Similarly, if customer i is visited at time t in s' but not in s, then the pair (i, t) is introduced in $\mathcal{L}_r$, and it is forbidden for some iterations to remove delivery time t at customer i. A pair (i, t) remains in $\mathcal{L}_a$ or $\mathcal{L}_r$ for $L + \text{random}(\lambda\sqrt{nR(s)})$ iterations, where L and λ are constants, n is the number of customers, $R(s)$ is the number of routes in s, and random(x) is an integer uniformly selected in the set $\{0, \ldots, \lfloor x \rfloor\}$, $\lfloor x \rfloor$ being the largest integer not greater than x. Such a random number is computed at each iteration and for each pair (i, t) introduced in $\mathcal{L}_a$ or $\mathcal{L}_r$. We use random numbers as in Taillard (1991) because preliminary tests showed that it gives better results than a fixed length.

As described below, a neighbor solution is obtained by adding and/or removing visits at some customers. These changes are performed as follows.

When we remove a visit to customer i at time t, we first remove customer i from the vehicle route at time t, and its predecessor is linked to its successor.

- In the case of the OU policy, the quantity delivered to i at time t is transferred to the following visit (if any). Such a removal is performed only if it creates no stockout at customer i to keep the solution admissible.
- For the ML policy, if there is no stockout at i when removing the visit, then nothing else is made; otherwise, the removal is performed only if the stockout at i can be avoided by increasing the quantity delivered at the previous visit (if any) to a value not larger than the maximum capacity U_i. More precisely, suppose x units are delivered to i at time t; let t' denote the time of the previous visit (if any) at i, and let y be the smallest value $I_{it''}(s)$ for $t'' > t$. If $y < x$, then the removal of the visit at time t creates a stockout for i, and we try to avoid it by delivering $x - y$ additional units at time t'. This is done if t' exists and if $I_{it'} + x_{it'} + (x - y) \le U_i$; otherwise, the removal is not performed.

When we insert a visit to customer i at time t, we first add i to the vehicle route at time t using the cheapest insertion method.

- In the case of the OU policy, the quantity delivered to i at time t is set equal to the difference between the maximum inventory level U_i and the current inventory level; the same quantity is removed from the next visit to i (if any).
- For the ML policy, the quantity delivered to i at time t is the minimum between the maximum quantity that can be delivered without exceeding the maximum capacity U_i, the residual capacity of the vehicle, and the quantity available at the supplier. If this minimum is equal to 0, then a quantity r_{it} (i.e., the demand at time t) is delivered to i, and this will create stockout at the supplier or a violation of the vehicle capacity constraint, but the solution will remain admissible.

The neighborhood $N(s)$ of s is created in two steps. We first build a set $N'(s)$ containing all admissible solutions that can be obtained from s with one of the following simple changes.

1. Remove a visit: Choose a customer i and a time $t \in T_i(s)$, and remove the visit to i at time t.

2. Insert a visit: Choose a customer i and a time $t \notin T_i(s)$, and add a new visit to i at time t.

3. Move a visit: Choose a customer i, times $t \in T_i(s)$ and $t' \notin T_i(s)$, remove the visit to i at time t, and add a new visit to i at time t'.

4. Swap two visits: Choose two customers i and j and two periods $t \in T_i(s) - T_j(s)$ and $t' \in T_j(s) - T_i(s)$, remove the visits to i at time t and to j at time t', and add new visits to i at time t' and to j at time t.

We then build $N(s)$ by applying additional changes on each solution in $N'(s)$. To explain these changes, consider any solution $s' \in N'(s)$. If $h_i > h_0$ for a customer i, then it might be interesting to remove visits to i. Indeed, such removals strictly decrease the total inventory cost, do not increase the transportation cost, and do not create any stockout at the supplier. Also, if the quantity $Q_t(s')$ delivered at time t is strictly larger than the vehicle capacity C or if there is a stockout at the supplier at time t (i.e., $B_t(s') < 0$), then the removal of a visit to a customer at time t strictly reduces the value of the penalty component in the objective function. Therefore, given a solution $s' \in N'(s)$, let $\mathcal{A}$ be the set of customers with different delivery times in s and s' (i.e., $T_i(s) \neq T_i(s')$). Choose a customer $i \in \mathcal{A}$. We consider all delivery times $t \in T_i(s')$ and all customers j delivered at time t in s' (i.e., all customers j such that $t \in T_j(s')$), and we test whether $h_j > h_0$, $Q_t(s') > C$, or $B_t(s') < 0$. If this is the case, then

- in the case of the OU policy, we remove the visit to j at time t if and only if this gives an admissible solution with a strict decrease of the objective function.
- in the case of the ML policy, we reduce the quantity delivered to j at time t as much as possible without creating a stockout (i.e., we remove $\min\{x_{jt}, \min_{t'>t} I_{jt'}\}$ units of delivery) if such a change gives a solution with a strict decrease of the objective function. If the delivery to j at time t is reduced to 0, we remove j from the route at time t.

If such a visit to j is removed, then j is inserted in $\mathcal{A}$, and the procedure is applied recursively to all delivery times $t \in T_j(s')$. Thus, set $\mathcal{A}$ is first formed by customer i used to build the set $N'(s)$. Once all $t \in T_i(s')$ are considered, customer i is removed from $\mathcal{A}$, and all customers j for which a visit has been removed are inserted in $\mathcal{A}$. Notice that if $s' \in N'(s)$ was obtained from s by inserting a delivery time t to customer i, then we do not consider the removal of the visit to i at time t. In other words, the removal of a visit in s' is permitted only if the visit also exists in s.

Moreover, in the case of the ML policy, we also consider all customers j served at time $t \in T_i(s')$ in s' and test whether $h_j < h_0$. In such a case it might be interesting to increase the quantity delivered to j at time t. We therefore increase the quantity delivered to j at time t as much as possible without exceeding the maximum inventory level (i.e., we add $U_j - \max_{t' \geq t}(I_{jt'} + x_{jt'})$ units of delivery) if such a change gives a solution with a strict decrease of the objective function. The construction of $N(s)$ is formally described in Algorithm 2, which can be found in the online supplement. All moves in $N(s)$ are evaluated. If a move is in $\mathcal{L}_a$ or in $\mathcal{L}_r$, then it is considered only if it leads to a solution that is better than the best solution found so far; otherwise, it is discarded. The procedure *Move* chooses the best solution in $N(s)$ as the next current solution. Once the best move is implemented, then the pair (i, t) is inserted in $\mathcal{L}_r$ in case we inserted a visit to customer i at time t, whereas it is inserted in $\mathcal{L}_a$ in case we removed a visit to customer i at time t. Note that the complexity of the procedure *Move* is $O(n^3H^3)$ as each visit to a customer is considered ($O(nH)$), and for each customer visit, at most $O(n^2H^2)$ operations have to be evaluated.

3.2. Improvement Phase

We have designed an *Improvement* procedure that tries to improve a given feasible solution by solving a sequence of MIPs. The next subsections give details on each of these MIPs, and we then prove that they all are NP-hard problems. We finally describe how to integrate these MIPs in HAIR.

3.2.1. Two MIPs. We first describe the considered MIPs in the case of the OU policy and then mention how to modify these MIPs for the ML policy.

In the first considered MIP, we try to improve a given solution s by assigning routes to time periods without changing the routes themselves but assigning them to possibly different time periods with respect to s. Thus, the customers visited by each route and the sequence of service remain the same, the only possible change being the removal of a customer from a route. In this case, the corresponding savings is calculated.

Let Δ_{ir} be the transportation savings gained if customer i is removed from route r (which, obviously, visits customer i in the incumbent solution). This savings is calculated by simply joining the predecessor with the successor of i. In the following, $\sigma_{ir} = 1$ if customer i is visited by route r in the incumbent solution. Let w_{ir} be a binary variable equal to 1 if customer i is removed from route r, θ_{it} be a binary variable equal to 1 if customer i is visited at time t (used to formulate the OU constraints), and z_{rt} be a binary variable equal to 1 if route r is assigned to time t. The MIP, which we call the *OU-route assignment problem* and denote by $MIP_1(OU, s)$, is formulated as follows:

$$\text{minimize} \quad \sum_{t \in \mathcal{T}'} h_0 B_t + \sum_{i \in \mathcal{M}} \sum_{t \in \mathcal{T}'} h_i I_{it} - \sum_{i \in \mathcal{M}} \sum_{r \in \mathcal{R}} \Delta_{ir} w_{ir} \tag{1}$$

$$\text{subject to} \quad B_t = B_{t-1} + r_{0t-1} - \sum_{i \in \mathcal{M}} x_{it-1}, \quad t \in \mathcal{T}', \tag{2}$$

$$B_t \geq \sum_{i \in \mathcal{M}} x_{it}, \quad t \in \mathcal{T}, \tag{3}$$

$$I_{it} = I_{it-1} + x_{it-1} - r_{it-1}, \quad i \in \mathcal{M}\ t \in \mathcal{T}', \tag{4}$$

$$x_{it} \geq U_i \theta_{it} - I_{it}, \quad i \in \mathcal{M}\ t \in \mathcal{T}, \tag{5}$$

$$x_{it} \leq U_i - I_{it}, \quad i \in \mathcal{M}\ t \in \mathcal{T}, \tag{6}$$

$$x_{it} \leq U_i \theta_{it}, \quad i \in \mathcal{M}\ t \in \mathcal{T}, \tag{7}$$

$$\sum_{i \in \mathcal{M}} x_{it} \leq C, \quad t \in \mathcal{T}, \tag{8}$$

$$\sum_{t \in \mathcal{T}} z_{rt} \leq 1, \quad r \in \mathcal{R}, \tag{9}$$

$$\sum_{r \in \mathcal{R}} z_{rt} \leq 1, \quad t \in \mathcal{T}, \tag{10}$$

$$x_{it} \leq U_i \sum_{r \in \mathcal{R}} (\sigma_{ir} z_{rt}), \quad i \in \mathcal{M}\ t \in \mathcal{T}, \tag{11}$$

$$x_{it} \leq U_i [2 - \sigma_{ir}(z_{rt} + w_{ir})], \quad i \in \mathcal{M}\ r \in \mathcal{R}\ t \in \mathcal{T}, \tag{12}$$

$$w_{ir} \leq \sigma_{ir} \sum_{t \in \mathcal{T}} z_{rt}, \quad i \in \mathcal{M}\ r \in \mathcal{R}, \tag{13}$$

$$x_{it} \geq 0, \quad i \in \mathcal{M}\ t \in \mathcal{T}, \tag{14}$$

$$I_{it} \geq 0, \quad i \in \mathcal{M}\ t \in \mathcal{T}', \tag{15}$$

$$B_t \geq 0, \quad t \in \mathcal{T}', \tag{16}$$

$$\theta_{it} \in \{0, 1\}, \quad i \in \mathcal{M}\ t \in \mathcal{T}, \tag{17}$$

$$w_{ir} \in \{0, 1\}, \quad i \in \mathcal{M}\ r \in \mathcal{R}, \tag{18}$$

$$z_{rt} \in \{0, 1\}, \quad r \in \mathcal{R}\ t \in \mathcal{T}, \tag{19}$$

where $\mathcal{R}$ is the set of routes in the given feasible solution s. The objective function includes the total inventory cost at supplier and customers and the total savings generated by the customers that are removed from routes where they are visited in the incumbent solution. Constraints (2)–(4) (where $r_{00} = 0$ and $x_{i0} = r_{i0} = 0$ for all $i \in \mathcal{M}$) define the inventory at the supplier and customers and avoid stockout at the supplier. Constraints (5)–(7) define the OU policy, and (8) is the vehicle capacity constraint. Constraints (9) establish that a route r can be assigned to at most one time period, and constraints (10) impose a maximum of one route at each time $t \in \mathcal{T}$. Constraints (11) impose that a customer can be served at time t only if it is visited by the route assigned at time t. Constraints (12) establish that a customer can not be served at time t if it is currently visited by the route assigned at time t but is removed from the route. Finally, constraints (13) impose that a customer can be removed only from a route where it is visited and only if this route is assigned to a time $t \in \mathcal{T}$.

Note that if more than one customer is removed from a route, then the savings component in the objective function is only an estimation of the real savings generated by the removal of the customers.

We do not consider the possibility of replicating a single route on different days (constraint (9)) because this created a remarkable increase in the complexity of the model and made the problem unsolvable in reasonable time. The same holds true for the case of allowing the insertion of a customer into a route.

The second MIP models a customer assignment problem. More precisely, given a solution s, we do not change the time assigned to each vehicle route in s, whereas the removal or insertion of customers into routes is allowed. Let Δ_{it} be the transportation savings gained if customer i is removed from the route assigned to time t (which, obviously, visits customer i in the incumbent solution). This savings is calculated by simply joining the predecessor and the successor of i. Let Γ_{it} be the insertion cost of customer i into the route at time t, calculated with the cheapest insertion method. In the following, $\sigma_{it} = 1$ if customer i is visited at time t in the incumbent solution. Let w_{it} be a binary variable equal to 1 if customer i is removed from the route at time t, and let v_{it} be a binary variable equal to 1 if customer i is inserted into the route at time t. Note that when compared with the previous MIP, the index t replaces the index r in Δ_{it}, σ_{it}, and w_{it} because the routes are now assigned to specific times. Moreover, the v_{it} are new variables needed because we now allow the insertion of customers into routes. In the case of the OU policy, the considered MIP is called the *OU-customer assignment problem* and is denoted by $MIP_2(OU, s)$. It contains variables x_{it}, I_{it}, B_t, and θ_{it} with the same meaning as in $MIP_1(OU, s)$. It is formulated as follows:

$$\text{minimize} \quad \sum_{t \in \mathcal{T}'} h_0 B_t + \sum_{i \in \mathcal{M}} \sum_{t \in \mathcal{T}'} h_i I_{it} - \sum_{i \in \mathcal{M}} \sum_{t \in \mathcal{T}} \Delta_{it} w_{it} + \sum_{i \in \mathcal{M}} \sum_{t \in \mathcal{T}} \Gamma_{it} v_{it} \tag{20}$$

subject to constraints (2)–(8), (14)–(17), and to

$$v_{it} \leq 1 - \sigma_{it}, \quad i \in \mathcal{M}\ t \in \mathcal{T}, \tag{21}$$

$$w_{it} \leq \sigma_{it}, \quad i \in \mathcal{M}\ t \in \mathcal{T}, \tag{22}$$

$$x_{it} \leq U_i(\sigma_{it} - w_{it} + v_{it}), \quad i \in \mathcal{M}\ t \in \mathcal{T}, \tag{23}$$

$$v_{it} \in \{0, 1\}, \quad i \in \mathcal{M}\ t \in \mathcal{T}, \tag{24}$$

$$w_{it} \in \{0, 1\}, \quad i \in \mathcal{M}\ t \in \mathcal{T}. \tag{25}$$

Constraints (21) establish that a customer cannot be inserted into a route that already visits it, whereas constraints (22) impose that a customer cannot be removed from a route that does not visit it. Finally, constraints (23) establish that a customer can be served at time t if it is visited by the route at time t and is not removed from it, or if it is inserted into the route at time t.

Again, if the optimal solution of $MIP_2(OU, s)$ contains a route that differs from the corresponding one

in s by at least two customers (i.e., at least two customers were removed or inserted, or one was removed and another one inserted), then the variation of the transportation cost in (20) is only an estimation of the real gain or loss induced by such a modification.

In the case of the ML policy, we consider the same MIPs as above, except that constraints (5), (7), and (17) disappear as they are only needed for the OU policy. The resulting MIPs are called the *ML-route assignment* and *ML-customer assignment* problems and are denoted by $MIP_1(ML, s)$ and $MIP_2(ML, s)$.

3.2.2. Complexity Results. In this section, we prove that all MIPs described in the previous section are NP-hard problems. We first use a reduction from the NP-hard *knapsack problem* (see Kellerer et al. 2004, Martello and Toth 1990) to prove that both the *OU-route assignment* and *OU-customer assignment* problems are NP-hard. The knapsack problem is defined as follows: given n objects of weight $w_i \in Z^+$ and value $v_i \in Z^+$ $(i = 1, \dots, n)$, and given a capacity $C < \sum_{i=1}^{n} w_i$, determine a subset A of $\{1, \dots, n\}$ with $\sum_{i \in A} w_i \le C$ and of maximum value $\sum_{i \in A} v_i$.

Note that the knapsack problem remains NP-hard if it is assumed that $\frac{2}{3}\sum_{i=1}^{n} w_i \le C < \sum_{i=1}^{n} w_i$. Indeed, given any instance I of the knapsack problem with n objects and $C < \frac{2}{3}\sum_{i=1}^{n} w_i$, one can build a new instance I' by adding an object $n+1$ of weight $w_{n+1} = 2\sum_{i=1}^{n} w_i - 3C + \delta$, $(0 \le \delta)$, and value $v_{n+1} = \sum_{i=1}^{n} v_i + 1$, and by considering a new capacity $C' = C + w_{n+1} = 2\sum_{i=1}^{n} w_i - 2C + \delta$. In this new instance it is always optimal to choose at least the new object because its value is strictly larger than the total value of the n other objects. Hence, once this object is chosen, the remaining capacity is $C' - w_{n+1} = C$, and the choice of the next objects is equivalent to the original instance I. Since $C < \sum_{i=1}^{n} w_i$, we have $C' = C + w_{n+1} < \sum_{i=1}^{n+1} w_i$. Also,

$$\begin{aligned} C' &= \frac{2}{3} \cdot 3\left(\sum_{i=1}^{n} w_i - C\right) + \delta \\ &= \frac{2}{3}\left(\sum_{i=1}^{n} w_i + \left(2\sum_{i=1}^{n} w_i - 3C + \delta\right)\right) + \frac{\delta}{3} \\ &= \frac{2}{3}\sum_{i=1}^{n+1} w_i + \frac{\delta}{3}. \end{aligned}$$

Hence, I' is as difficult as I and $\frac{2}{3}\sum_{i=1}^{n+1} w_i \le C' < \sum_{i=1}^{n+1} w_i$.

THEOREM 1. *The OU-route assignment and OU-customer assignment problems are both NP-hard.*

PROOF. Given an instance of the knapsack problem with n objects of weight w_i and size v_i $(i = 1, \dots, n)$ and with capacity C such that $\frac{2}{3}\sum_{i=1}^{n} w_i \le C < \sum_{i=1}^{n} w_i$, we construct an instance of the IRP where $\mathcal{M} = \{1, \dots, n\}$, $\mathcal{T} = \{1, 2\}$, $r_{0t} = 0$ for $t \in \mathcal{T}$, $r_{it} = r_i = w_i/3$ for $i \in \mathcal{M}$ and $t \in \mathcal{T}$, $B_0 = \sum_{i \in M} 3r_i$, $h_0 = 0$, $I_{i0} = r_i$, $U_i = 3r_i$, $h_i = v_i/r_i > 0$ for all customers i, and a transportation capacity C. The above assumption about C translates to $\sum_{i \in \mathcal{M}} 2r_i \le C < \sum_{i \in \mathcal{M}} 3r_i$. Assume that we are given a solution s that contains two routes, each of them visiting all the customers, and suppose that $\Delta_{i1} = \Delta_{i2} = \Delta_i > 0$ for all $i \in \mathcal{M}$.

All customers have to be served at least once because the initial inventory level $I_{i0} = r_i$ is not sufficient for the total time horizon. It is not necessary to serve a customer at time $t = 1$ because the initial inventory I_{i0} equals r_i. If a customer i is served at time $t = 1$, then $I_{i2} = 2r_i$, and it is therefore not necessary to make a second delivery at time $t = 2$.

Hence, in all optimal solutions, every customer is visited exactly once. If a customer i is visited at time $t = 1$, then $I_{i1} = r_i$, $I_{i2} = 2r_i$, and $I_{i3} = r_i$, which gives a total inventory of $4r_i$, whereas if i is visited at time $t = 2$, then $I_{i1} = r_i$, $I_{i2} = 0$, and $I_{i3} = 2r_i$, for a total inventory of $3r_i$. The starting inventory level $B_0 = \sum_{i \in M} 3r_i$ at the supplier indicates that no solution visiting each customer exactly once creates a stockout at the supplier. Since $h_0 = 0$, the supplier has no preference between deliveries at time $t = 1$ or 2. However, since $h_i > 0$ for all $i \in \mathcal{M}$, each customer prefers to be served at time $t = 2$.

The transportation capacity C forces some of the customers to be served at time $t = 1$. Note that the vehicle has enough capacity to serve all customers at time $t = 1$ since the vehicle load would be $\sum_{i \in \mathcal{M}} 2r_i \le C$ in that case. Because no customer needs to be served twice, each one will be removed from exactly one route, which gives a saving Δ_i that does not depend on the time t. Because both routes in s contain all customers, no customer can be added to a route. Hence, the objective functions in (1) and (20) can be rewritten as follows, where $\mathcal{M}_t$ denotes the set of customers served at time t:

$$\sum_{i \in \mathcal{M}_1} 4h_i r_i + \sum_{i \in \mathcal{M}_2} 3h_i r_i - \sum_{i \in \mathcal{M}} \Delta_i = \sum_{i \in \mathcal{M}} (4h_i r_i - \Delta_i) - \sum_{i \in \mathcal{M}_2} h_i r_i.$$

Because the first term is a constant, this is equivalent to maximizing $\sum_{i \in \mathcal{M}_2} h_i r_i$. If we introduce a binary variable y_i that takes a value of 1 if customer i is removed from the route used at time $t = 1$ and 0 otherwise, the *OU-route assignment* and *OU-customer assignment* problems can be equivalently formulated as follows:

$$\begin{aligned} \text{maximize} \quad & \sum_{i \in \mathcal{M}} h_i r_i y_i \\ \text{subject to} \quad & \sum_{i \in \mathcal{M}} 3r_i y_i \le C, \\ & y_i \in \{0, 1\}, \quad i \in \mathcal{M}. \end{aligned}$$

Because we have defined h_i and r_i so that $h_i r_i = v_i$ and $3r_i = w_i$, the above problem is equivalent to the considered knapsack problem. □

To prove that both *ML-route assignment* and *ML-customer assignment* are NP-hard problems, we use a reduction from the NP-complete *partition problem* (see Garey and Johnson 1979), which is defined as follows: given n objects of weight $w_i \in Z^+$, determine whether there exists a subset A of $\{1, \ldots, n\}$ with $\sum_{i \in A} w_i = \sum_{i \notin A} w_i$.

THEOREM 2. *The ML-route assignment and ML-customer assignment problems are both NP-hard.*

PROOF. Given an instance of the partition problem with n objects of size w_i $(i = 1, \ldots, n)$, we consider an instance of the IRP with $\mathcal{M} = \{1, \ldots, n\}$, $\mathcal{T} = \{1, 2\}$, $r_{0t} = 0$ for $t \in \mathcal{T}$, $r_{it} = w_i$ for $i \in \mathcal{M}$, and $t \in \mathcal{T}$, $B_0 = \sum_{i \in \mathcal{M}} w_i$, $h_0 = 0$, $I_{i0} = w_i$, $U_i = 2w_i$, $h_i = 1$ for all customers i, and a transportation capacity $C = \frac{1}{2} \sum_{i \in \mathcal{M}} w_i$. Assume that we are given a solution s that contains two routes, each of them delivering $(1/2)w_i$ units to each customer i. Suppose finally that $\Delta_{i1} = \Delta_{i2} = 2C$ for all $i \in \mathcal{M}$.

All customers have to be served at least once because the initial inventory level $I_{i0} = w_i$ is not sufficient for the total time horizon. It is not necessary to serve a customer at time $t = 1$ because the initial inventory I_{i0} equals w_i. It is never optimal to send to a customer i a total quantity greater than w_i. Therefore, $0 \le x_{i1} \le w_i$ and $x_{i2} = w_i - x_{i1}$. The inventory levels at i are $I_{i1} = w_i$, $I_{i2} = x_{i1}$, and $I_{i3} = 0$, which gives a total inventory cost of $w_i + x_{i1}$.

Because both routes in s contain all customers, no customer can be added to a route. Hence, since $h_0 = 0$, both (1) and (20) reduce to the difference between the second and the third term, that is, $\sum_{i \in \mathcal{M}} (w_i + x_{i1}) - 2Cm$, where m is the number of customers removed.

If a solution has at least one customer that is visited in periods $t = 1$ and $t = 2$, then $\sum_{i \in \mathcal{M}} (w_i + x_{i1}) - 2Cm > \sum_{i \in \mathcal{M}} w_i - 2C(n-1) = 2C(2-n)$, whereas the value of a solution in which no customer is visited twice is strictly smaller than $\sum_{i \in \mathcal{M}} 2w_i - 2Cn = 2C(2-n)$. Hence, the optimal value of the *ML-route assignment* and *ML-customer assignment* problems is strictly smaller than $2C(2-n)$ if and only if there is a partition of the customers into two sets $\mathcal{M}_1$ and $\mathcal{M}_2$ with $\sum_{i \in \mathcal{M}_1} w_i \le C$ and $\sum_{i \in \mathcal{M}_2} w_i \le C$. Because $C = \frac{1}{2} \sum_{i \in \mathcal{M}} w_i$, this is equivalent to saying that $\sum_{i \in \mathcal{M}_1} w_i = \sum_{i \in \mathcal{M}_2} w_i$, which means that the partition problem has a solution. □

3.2.3. The Improvement Procedure. Each time s_{best} is improved by our tabu search, we try to generate a better solution by first considering every route in the solution and trying to reduce the transportation cost by applying the heuristic by Lin and Kernighan (1973) for the TSP. It is known that the Lin and Kernighan heuristic is very effective in finding near-optimal solutions for TSP instances, especially for small-size instances. In our case, the maximum number of customers contained in a route is equal to n, and in our tests we often find solutions where there is at least a route visiting all, or almost all, customers. However, because the maximum size of the instances we tested is $n = 200$, the Lin and Kernighan heuristic always finds an optimal or almost optimal TSP tour. We then apply to the new solution the MIPs described in §3.2.1. If the output of one of these MIPs differs from the input, we apply again the Lin and Kernighan heuristic on each route that was modified. We first try to improve s_{best} by solving $MIP_1(\diamond, s_{\text{best}})$ (with $\diamond =$ OU or ML). We then consider all pairs (r_1, r_2) of consecutive routes in s_{best} (i.e., routes assigned to two consecutive time periods). For each such pair, we replace the two routes by a single "big route" r containing all customers of r_1 and r_2. Such a big route is obtained by inserting in r_1 all customers that are in r_2 but not in r_1 (using the cheapest insertion method) and by then optimizing the route using the Lin and Kernighan heuristic. Let s_i $(i = 1, 2)$ be the solution obtained from s_{best} by removing r_1 and r_2 and by assigning route r to the same time as r_i. We then solve $MIP_2(\diamond, s_i)$ $(i = 1, 2, \diamond =$ OU or ML). It may happen that one or both of these problems has no feasible solution because a stockout at the customers is possibly unavoidable after the merge of r_1 and r_2 into a single route r. If $MIP_2(\diamond, s_1)$ is infeasible, we modify s_1 by taking the first route after r and anticipating it by one time period, and we then solve $MIP_2(\diamond, s_1)$ again. Similarly, if $MIP_2(\diamond, s_2)$ is infeasible, we take the first route that precedes r in s_2, delay it by one time period, and solve $MIP_2(\diamond, s_2)$ again. This process is repeated for each pair (r_1, r_2) of consecutive routes, and the best obtained solution replaces s_{best} if its value is strictly smaller than $f(s_{\text{best}})$. We finally try to improve s_{best} by solving $MIP_2(\diamond, s_{\text{best}})$. A detailed description of the *Improvement* procedure can be found in Algorithm 3 in the online supplement.

3.3. Jump Procedure

When the number of iterations without improvement of s_{best} is a multiple of a given parameter *JumpIter*, a jump from s_{best} to a new solution is made that consists in moving customers from time periods where they are typically visited to time periods where they are typically not visited. More precisely, as long as there exists a triplet (i, t, t') such that i is a customer visited at time t since at least *JumpIter*/2 iterations, i was never visited at time $t' \neq t$ during the last *JumpIter*/2 iterations, and the move of the visit to i from t to t' does not create a stockout at i, then we choose such a triplet at random and perform the move of the visit to

i from t to t'. When no additional changes of this type can be done, we apply the second MIP of the *Improvement* procedure (see §3.2) and consider the resulting solution as the new current solution s for the tabu search.

4. Computational Results

The hybrid heuristic HAIR has been implemented in C++ and run on an Intel Dual Core 1.86 GHz and 3.2 GB RAM personal computer. The *route assignment* problem (MIP_1) and the *customer assignment* problem (MIP_2) have been solved to optimality by using CPLEX 10.1.

The hybrid heuristic HAIR was tested on the benchmark instances used in Archetti et al. (2007), which have the following characteristics.

- Time horizon H: 3, 6.
- Number of customers n: $5k$, with $k = 1, 2, \ldots, 10$ when $H = 3$, and $k = 1, 2, \ldots, 6$ when $H = 6$.
- Product quantity r_{it} consumed by customer i at time t: constant over time, i.e., $r_{it} = r_i$, $t \in \mathcal{T}$, and randomly generated as an integer number in the interval [10, 100].
- Product quantity r_{0t} made available at the supplier at time t: $\sum_{i \in \mathcal{M}} r_i$.
- Maximum inventory level U_i at customer i: $r_i g_i$, where g_i is randomly selected from the set {2, 3} and represents the number of time units needed to consume the quantity U_i.
- Starting inventory level B_0 at the supplier: $\sum_{i \in \mathcal{M}} U_i$.
- Starting inventory level I_{i0} at the customer i: $U_i - r_i$.
- Inventory cost at customer $i \in \mathcal{M}$, h_i: randomly generated in the intervals [0.01, 0.05] and [0.1, 0.5];
- Inventory cost at the supplier h_0: 0.03 when h_i is generated in [0.01, 0.05] and 0.3 when h_i is generated in [0.1, 0.5].
- Transportation capacity C: $(3/2) \sum_{i \in \mathcal{M}} r_i$.
- Transportation cost c_{ij}: $\lfloor \sqrt{(X_i - X_j)^2 + (Y_i - Y_j)^2} \rfloor$, where the points (X_i, Y_i) and (X_j, Y_j) are obtained by randomly generating each coordinate as an integer number in the interval [0, 500].

For each combination of n, H, an interval for the generation of h_i, and h_0, five instances were tested for a total of 160 instances. The values of the parameters are as follows: $MaxIter = 200nH$, $JumpIter = MaxIter/2$, and $L = 10$, whereas the value of λ depends on the instance and is specified in Tables 1–8 (and 13–16). Moreover, on the basis of preliminary tests, we decided to apply the *Jump* procedure only once.

A local search heuristic for the case with OU policy was proposed in Bertazzi et al. (2002). In the following, we will also report the performance of this heuristic. The results are shown in Tables 1–4 for the ML policy and in Tables 5–8 for the OU policy. In these tables, column 1 reports the name of the instance, "n" is the number of customers, and "λ" is the parameter (in addition to $L = 10$) used to deter-

Table 1 **ML Policy—Computational Results with $H = 3$ and $h_i \in [0.01, 0.05]$**

Instance	n	λ	z^*	HAIR	CPU	% error
abs1n5.dat	5	0.5	1,235.92	1,235.92	2	0.00
abs2n5.dat	5	0.5	988.66	988.66	2	0.00
abs3n5.dat	5	0.5	1,758.02	1,758.02	3	0.00
abs4n5.dat	5	0.5	1,397.29	1,397.29	4	0.00
abs5n5.dat	5	0.5	999.42	999.42	2	0.00
abs1n10.dat	10	0.5	1,743.07	1,743.07	10	0.00
abs2n10.dat	10	0.5	2,229.25	2,229.25	10	0.00
abs3n10.dat	10	0.5	1,871.14	1,871.14	9	0.00
abs4n10.dat	10	0.5	1,773.00	1,773.00	8	0.00
abs5n10.dat	10	0.5	1,938.18	1,938.18	10	0.00
abs1n15.dat	15	0.5	2,131.04	2,131.04	30	0.00
abs2n15.dat	15	0.5	2,131.58	2,131.58	24	0.00
abs3n15.dat	15	0.5	2,463.68	2,463.68	25	0.00
abs4n15.dat	15	0.5	2,151.94	2,151.94	34	0.00
abs5n15.dat	15	0.5	2,160.59	2,160.59	21	0.00
abs1n20.dat	20	0.5	2,267.32	2,267.32	43	0.00
abs2n20.dat	20	0.5	2,497.90	2,497.90	72	0.00
abs3n20.dat	20	0.5	2,590.48	2,590.48	60	0.00
abs4n20.dat	20	0.5	3,122.31	3,122.99	90	0.02
abs5n20.dat	20	0.5	2,849.90	2,849.90	58	0.00
abs1n25.dat	25	0.5	2,840.92	2,840.92	107	0.00
abs2n25.dat	25	0.5	3,014.56	3,014.56	92	0.00
abs3n25.dat	25	0.5	3,050.40	3,050.40	100	0.00
abs4n25.dat	25	0.5	3,078.67	3,078.67	161	0.00
abs5n25.dat	25	0.5	2,954.96	2,954.96	104	0.00
abs1n30.dat	30	0.5	3,427.78	3,427.78	221	0.00
abs2n30.dat	30	0.5	3,328.94	3,328.94	190	0.00
abs3n30.dat	30	0.5	3,471.86	3,471.86	249	0.00
abs4n30.dat	30	0.5	3,321.48	3,321.48	436	0.00
abs5n30.dat	30	0.5	2,914.60	2,914.60	201	0.00
abs1n35.dat	35	0.5	3,346.12	3,346.12	446	0.00
abs2n35.dat	35	0.5	3,541.71	3,541.71	601	0.00
abs3n35.dat	35	0.5	3,811.78	3,811.78	340	0.00
abs4n35.dat	35	0.5	3,229.34	3,229.34	402	0.00
abs5n35.dat	35	0.5	3,315.26	3,315.26	284	0.00
abs1n40.dat	40	0.5	3,702.14	3,702.14	589	0.00
abs2n40.dat	40	0.5	3,832.09	3,832.09	546	0.00
abs3n40.dat	40	0.5	3,874.62	3,874.62	471	0.00
abs4n40.dat	40	0.5	3,534.80	3,534.80	550	0.00
abs5n40.dat	40	0.5	3,575.46	3,575.46	450	0.00
abs1n45.dat	45	0.5	3,950.86	3,950.86	939	0.00
abs2n45.dat	45	0.5	3,702.72	3,702.72	700	0.00
abs3n45.dat	45	0.5	3,968.04	3,968.04	668	0.00
abs4n45.dat	45	0.5	3,998.26	3,998.26	837	0.00
abs5n45.dat	45	0.5	3,717.54	3,717.54	744	0.00
abs1n50.dat	50	0.5	4,047.18	4,047.18	1,009	0.00
abs2n50.dat	50	0.5	4,512.96	4,512.96	1,229	0.00
abs3n50.dat	50	0.5	4,451.44	4,451.44	1,112	0.00
abs4n50.dat	50	0.5	4,405.84	4,405.84	1,105	0.00
abs5n50.dat	50	0.5	4,218.37	4,218.37	982	0.00

Table 2 **ML Policy—Computational Results with $H = 3$ and $h_i \in [0.1, 0.5]$**

Instance	n	λ	z^*	HAIR	CPU	% error
abs1n5.dat	5	0.5	2,108.34	2,108.34	1	0.00
abs2n5.dat	5	0.5	1,767.06	1,767.06	1	0.00
abs3n5.dat	5	0.5	2,973.00	2,973.00	3	0.00
abs4n5.dat	5	0.5	1,981.04	1,981.04	4	0.00
abs5n5.dat	5	0.5	2,170.04	2,170.04	1	0.00
abs1n10.dat	10	0.5	4,510.61	4,510.61	10	0.00
abs2n10.dat	10	0.5	4,504.61	4,504.61	11	0.00
abs3n10.dat	10	0.5	4,031.40	4,031.40	9	0.00
abs4n10.dat	10	0.5	3,933.46	3,933.46	9	0.00
abs5n10.dat	10	0.5	4,709.79	4,709.79	10	0.00
abs1n15.dat	15	0.5	5,589.70	5,589.70	31	0.00
abs2n15.dat	15	0.5	5,443.34	5,443.34	38	0.00
abs3n15.dat	15	0.5	6,300.86	6,300.86	27	0.00
abs4n15.dat	15	0.5	4,977.58	4,977.58	44	0.00
abs5n15.dat	15	0.5	4,867.53	4,867.53	21	0.00
abs1n20.dat	20	0.5	6,859.02	6,859.02	44	0.00
abs2n20.dat	20	0.5	7,087.74	7,087.74	76	0.00
abs3n20.dat	20	0.5	7,354.68	7,354.68	68	0.00
abs4n20.dat	20	0.5	6,952.79	6,957.78	88	0.07
abs5n20.dat	20	0.5	7,874.26	7,874.26	57	0.00
abs1n25.dat	25	0.5	8,227.86	8,227.86	115	0.00
abs2n25.dat	25	0.5	8,765.72	8,765.72	102	0.00
abs3n25.dat	25	0.5	9,382.42	9,382.42	108	0.00
abs4n25.dat	25	0.5	8,452.93	8,452.93	165	0.00
abs5n25.dat	25	0.5	10,081.42	10,081.42	116	0.00
abs1n30.dat	30	0.5	12,066.86	12,066.86	234	0.00
abs2n30.dat	30	0.5	10,941.32	10,941.32	271	0.00
abs3n30.dat	30	0.5	12,122.36	12,122.36	245	0.00
abs4n30.dat	30	0.5	9,687.10	9,687.10	312	0.00
abs5n30.dat	30	0.5	9,773.90	9,773.90	209	0.00
abs1n35.dat	35	0.5	11,659.88	11,659.88	483	0.00
abs2n35.dat	35	0.5	10,466.80	10,466.80	448	0.00
abs3n35.dat	35	0.5	13,776.46	13,776.46	322	0.00
abs4n35.dat	35	0.5	10,307.40	10,307.40	447	0.00
abs5n35.dat	35	0.5	10,847.82	10,847.82	297	0.00
abs1n40.dat	40	0.5	13,364.92	13,364.92	624	0.00
abs2n40.dat	40	0.5	11,317.85	11,317.85	612	0.00
abs3n40.dat	40	0.5	13,598.94	13,598.94	544	0.00
abs4n40.dat	40	0.5	11,353.39	11,353.39	593	0.00
abs5n40.dat	40	0.5	13,070.18	13,070.18	523	0.00
abs1n45.dat	45	0.5	14,179.10	14,179.10	958	0.00
abs2n45.dat	45	0.5	13,142.22	13,142.22	746	0.00
abs3n45.dat	45	0.5	14,843.60	14,843.60	717	0.00
abs4n45.dat	45	0.5	13,574.50	13,574.50	876	0.00
abs5n45.dat	45	0.5	13,587.26	13,587.26	732	0.00
abs1n50.dat	50	0.5	14,577.30	14,577.30	1,092	0.00
abs2n50.dat	50	0.5	15,001.64	15,001.64	1,036	0.00
abs3n50.dat	50	0.5	15,279.49	15,279.49	1,342	0.00
abs4n50.dat	50	0.5	16,517.00	16,517.00	1,217	0.00
abs5n50.dat	50	0.5	15,678.67	15,678.67	1,041	0.00

mine for how long an element remains in a tabu list. The value of the optimal solution "z^*" is taken from Archetti et al. (2007). Column HAIR reports the value of the solution given by HAIR, "CPU" is the computational time in seconds, and "% error" gives the percentage error of the HAIR solution with respect to the optimal solution. For the OU policy, we have also reported the results given by the heuristic algorithm in Bertazzi et al. (2002): column "BPS" reports the solution value, and "% error" indicates the error with respect to the optimal solution. Note that because of the randomness of the length of the tabu lists, different runs of HAIR lead to possibly different solutions. However, we made some preliminary tests with different seeds of the random numbers generator, and we observed that we rarely obtained different solutions. When this happens, the difference between the values of the objective function was always less than 0.1%. Thus, we decided to make only one run of HAIR in the following tests.

Table 3 **ML Policy—Computational Results with $H = 6$ and $h_i \in [0.01, 0.05]$**

Instance	n	λ	z^*	HAIR	CPU	% error
abs1n5.dat	5	0.5	3,187.30	3,187.3	17	0.00
abs2n5.dat	5	0.5	2,565.92	2,566.47	19	0.02
abs3n5.dat	5	0.5	4,489.83	4,489.83	19	0.00
abs4n5.dat	5	0.5	3,174.35	3,174.35	14	0.00
abs5n5.dat	5	0.5	2,267.10	2,267.1	11	0.00
abs1n10.dat	10	0.5	4,141.53	4,141.53	69	0.00
abs2n10.dat	10	0.5	5,044.63	5,046.13	69	0.03
abs3n10.dat	10	0.5	4,506.83	4,508.8	71	0.04
abs4n10.dat	10	0.5	4,823.53	4,823.53	49	0.00
abs5n10.dat	10	0.5	4,545.98	4,546.03	65	0.00
abs1n15.dat	15	0.5	5,389.08	5,391.17	166	0.04
abs2n15.dat	15	0.5	5,418.47	5,423.47	245	0.09
abs3n15.dat	15	0.5	5,897.68	5,897.68	161	0.00
abs4n15.dat	15	0.5	5,335.01	5,335.01	203	0.00
abs5n15.dat	15	0.5	5,052.51	5,074.68	127	0.44
abs1n20.dat	20	0.5	6,114.04	6,114.04	475	0.00
abs2n20.dat	20	0.5	5,957.31	5,958.29	561	0.02
abs3n20.dat	20	0.5	6,784.06	6,786.04	429	0.03
abs4n20.dat	20	0.5	7,309.54	7,394.84	508	1.17
abs5n20.dat	20	0.5	6,961.82	6,967.4	309	0.08
abs1n25.dat	25	0.5	7,052.06	7,064	1,678	0.16
abs2n25.dat	25	0.5	7,231.75	7,254.38	725	0.31
abs3n25.dat	25	0.5	7,514.57	7,514.57	1,162	0.00
abs4n25.dat	25	0.5	7,462.08	7,462.08	744	0.00
abs5n25.dat	25	0.5	7,048.40	7,059.9	845	0.16
abs1n30.dat	30	0.5	8,052.73	8,092.11	1,930	0.49
abs2n30.dat	30	0.5	7,629.99	7,631.21	2,602	0.02
abs3n30.dat	30	0.5	8,136.21	8,137.33	2,422	0.01
abs4n30.dat	30	0.5	7,502.49	7,502.49	3,712	0.00
abs5n30.dat	30	0.5	7,228.63	7,265.35	2,003	0.51

The results prove the effectiveness of HAIR both for the OU and for the ML policy. The average error is 0.08% for the former policy and 0.05% for the latter policy with individual errors only twice above 1%. The class of instances where HAIR is most successful is the class with short horizon ($H = 3$) for the ML policy (see Tables 1 and 2). Both when $h_i \in [0.1, 0.5]$ and

Table 4 **ML Policy—Computational Results with $H = 6$ and $h_i \in [0.1, 0.5]$**

Instance	n	λ	z^*	HAIR	CPU	% error
abs1n5.dat	5	0.5	5,789.35	5,789.35	16	0.00
abs2n5.dat	5	0.5	4,883.77	4,883.77	14	0.00
abs3n5.dat	5	0.5	6,643.29	6,643.29	24	0.00
abs4n5.dat	5	0.5	5,076.88	5,076.88	22	0.00
abs5n5.dat	5	0.5	4,377.71	4,377.71	13	0.00
abs1n10.dat	10	0.5	8,480.17	8,480.17	86	0.00
abs2n10.dat	10	0.5	8,347.44	8,356.35	79	0.11
abs3n10.dat	10	0.5	8,321.68	8,360.05	107	0.46
abs4n10.dat	10	0.5	8,474.26	8,474.26	77	0.00
abs5n10.dat	10	0.5	9,386.03	9,386.03	82	0.00
abs1n15.dat	15	0.5	12,052.56	12,052.56	179	0.00
abs2n15.dat	15	0.5	11,823.55	11,825.87	243	0.02
abs3n15.dat	15	0.5	13,305.71	13,305.71	270	0.00
abs4n15.dat	15	0.5	10,479.29	10,494.11	239	0.14
abs5n15.dat	15	0.5	10,054.09	10,070.56	201	0.16
abs1n20.dat	20	0.5	14,266.53	14,301	450	0.24
abs2n20.dat	20	0.5	14,477.84	14,551.58	392	0.51
abs3n20.dat	20	0.5	14,319.35	14,328.68	426	0.07
abs4n20.dat	20	0.5	14,390.27	14,417	398	0.19
abs5n20.dat	20	0.5	15,556.70	15,563.44	341	0.04
abs1n25.dat	25	0.5	15,487.66	15,555.79	805	0.44
abs2n25.dat	25	0.5	16,502.46	16,516.45	795	0.08
abs3n25.dat	25	0.5	17,833.35	17,833.35	894	0.00
abs4n25.dat	25	0.5	16,193.96	16,316.37	1,070	0.76
abs5n25.dat	25	0.5	18,552.42	18,570.92	1,570	0.10
abs1n30.dat	30	0.5	22,837.94	22,874.4	1,922	0.16
abs2n30.dat	30	0.5	19,876.76	19,997.28	1,850	0.61
abs3n30.dat	30	0.5	23,096.93	23,198.29	2,192	0.44
abs4n30.dat	30	0.5	17,509.82	17,550.33	1,952	0.23
abs5n30.dat	30	0.5	18,731.81	18,770.28	1,694	0.21

when $h_i \in [0.01, 0.05]$, HAIR finds all optimal solutions, except in one case (abs4n20.dat). In general, in 88% of the instances with short horizon $H = 3$ HAIR finds the optimal solution, whereas the optimal solution is found on 52.5% of the instances with horizon $H = 6$.

Some tests were run to evaluate the impact of the improvement phase on the quality of the solution. Five instances with $n = 30$, $H = 6$, $h_i \in [0.1, 0.5]$, and $h_0 = 0.3$ were chosen for these tests. The results are shown in Tables 9–12. The first two tables refer to the case where the algorithm is stopped after *MaxIter* iterations without improvement, and the second two tables refer to the case where the algorithm is stopped after 10,000 iterations. The first column reports the name of the instance. Then, for each algorithm, two columns report the error with respect to the optimal solution and the CPU time, respectively. In the following columns, "HAIR" stands for the complete hybrid algorithm, "Tabu" is HAIR without any improvement phase, and the remaining columns report the errors obtained with a partial improvement phase: "1" means that MIP_1 is applied on s_{best},

Table 5 **OU Policy—Computational Results with $H = 3$ and $h_i \in [0.01, 0.05]$**

Instance	n	λ	z^*	HAIR	CPU	% error	BPS	% error
abs1n5.dat	5	0.5	1,281.68	1,281.68	3	0.00	1,281.68	0.00
abs2n5.dat	5	0.5	1,176.63	1,176.63	3	0.00	1,176.63	0.00
abs3n5.dat	5	0.5	2,020.65	2,020.65	4	0.00	2,178.94	7.83
abs4n5.dat	5	0.5	1,449.43	1,449.43	2	0.00	1,449.43	0.00
abs5n5.dat	5	0.5	1,165.40	1,165.40	3	0.00	1,242.07	6.58
abs1n10.dat	10	0.5	2,167.37	2,167.37	14	0.00	2,198.56	1.44
abs2n10.dat	10	0.5	2,510.13	2,510.13	12	0.00	2,510.13	0.00
abs3n10.dat	10	0.5	2,099.68	2,099.68	13	0.00	2,099.68	0.00
abs4n10.dat	10	0.5	2,188.01	2,188.01	11	0.00	2,188.01	0.00
abs5n10.dat	10	0.5	2,178.15	2,178.45	14	0.01	2,231.67	2.46
abs1n15.dat	15	0.5	2,236.53	2,236.53	40	0.00	2,271.68	1.57
abs2n15.dat	15	0.5	2,506.21	2,506.21	41	0.00	2,506.21	0.00
abs3n15.dat	15	0.5	2,841.06	2,841.06	44	0.00	2,841.06	0.00
abs4n15.dat	15	0.5	2,430.07	2,430.07	43	0.00	2,632.18	8.32
abs5n15.dat	15	0.5	2,453.50	2,453.50	39	0.00	2,524.95	2.91
abs1n20.dat	20	0.5	2,793.29	2,793.29	75	0.00	2,793.29	0.00
abs2n20.dat	20	0.5	2,799.90	2,799.90	105	0.00	2,873.51	2.63
abs3n20.dat	20	0.5	3,101.60	3,104.29	101	0.09	3,163.94	2.01
abs4n20.dat	20	0.5	3,239.31	3,239.31	87	0.00	3,239.31	0.00
abs5n20.dat	20	0.5	3,330.99	3,330.99	153	0.00	3,814.54	14.52
abs1n25.dat	25	0.5	3,309.64	3,309.64	341	0.00	3,624.49	9.51
abs2n25.dat	25	0.5	3,495.97	3,495.97	214	0.00	3,544.55	1.39
abs3n25.dat	25	0.5	3,481.45	3,481.45	278	0.00	3,622.70	4.06
abs4n25.dat	25	0.5	3,272.74	3,272.74	295	0.00	3,272.74	0.00
abs5n25.dat	25	0.5	3,695.94	3,695.94	166	0.00	3,695.94	0.00
abs1n30.dat	30	0.5	3,918.76	3,918.76	326	0.00	4,022.30	2.64
abs2n30.dat	30	0.5	3,737.11	3,737.11	497	0.00	3,874.22	3.67
abs3n30.dat	30	0.5	3,761.85	3,761.85	607	0.00	3,931.85	4.52
abs4n30.dat	30	0.5	3,532.47	3,532.47	452	0.00	3,676.90	4.09
abs5n30.dat	30	0.5	3,265.89	3,269.76	693	0.12	3,365.76	3.06
abs1n35.dat	35	0.5	3,694.48	3,694.48	908	0.00	3,737.48	1.16
abs2n35.dat	35	0.5	3,796.80	3,796.80	704	0.00	3,960.39	4.31
abs3n35.dat	35	0.5	4,351.09	4,359.08	532	0.18	4,665.40	7.22
abs4n35.dat	35	0.5	3,766.39	3,766.39	1,225	0.00	3,939.11	4.59
abs5n35.dat	35	0.5	3,625.57	3,625.57	675	0.00	3,807.86	5.03
abs1n40.dat	40	0.5	4,263.43	4,274.71	903	0.26	4,732.75	11.01
abs2n40.dat	40	0.5	4,166.95	4,166.95	1,539	0.00	4,415.23	5.96
abs3n40.dat	40	0.5	4,337.30	4,340.06	1,310	0.06	4,591.92	5.87
abs4n40.dat	40	0.5	3,846.84	3,846.84	1,441	0.00	4,000.81	4.00
abs5n40.dat	40	0.5	4,013.98	4,013.98	650	0.00	4,234.01	5.48
abs1n45.dat	45	0.5	4,369.38	4,369.38	1,493	0.00	4,568.68	4.56
abs2n45.dat	45	0.5	4,226.82	4,226.82	1,224	0.00	4,655.80	10.15
abs3n45.dat	45	0.5	4,317.08	4,317.08	1,904	0.00	4,572.82	5.92
abs4n45.dat	45	0.5	4,527.95	4,559.36	1,292	0.69	4,962.14	9.59
abs5n45.dat	45	0.5	3,911.82	3,911.82	1,387	0.00	4,215.11	7.75
abs1n50.dat	50	1	4,629.92	4,670.41	1,782	0.87	4,884.83	5.51
abs2n50.dat	50	1	4,919.75	4,919.75	2,004	0.00	5,123.98	4.15
abs3n50.dat	50	1	4,868.36	4,868.36	2,648	0.00	5,269.43	8.24
abs4n50.dat	50	1	4,972.25	5,014.17	1,843	0.84	5,273.98	6.07
abs5n50.dat	50	1	4,664.05	4,687.16	3,126	0.50	4,901.19	5.08

"2" means that MIP_2 is applied on solutions obtained from s_{best} by merging a pair of routes assigned to two consecutive time periods (see §3.2.3), and "3" means that MIP_2 is applied on s_{best}. The results are shown for any combination of possible improvements and show the importance of combining the tabu search scheme with optimization models that effectively explore the solution space. From Tables 9 and 10, it can be observed that in most cases HAIR finds better solu-

Table 6 OU Policy—Computational Results with $H = 3$ and $h_i \in [0.1, 0.5]$

Instance	n	λ	z^*	HAIR	CPU	% error	BPS	% error
abs1n5.dat	5	0.5	2,149.80	2,149.80	4	0.00	2,149.80	0.00
abs2n5.dat	5	0.5	1,959.05	1,959.05	4	0.00	1,959.05	0.00
abs3n5.dat	5	0.5	3,265.44	3,265.44	7	0.00	3,408.48	4.38
abs4n5.dat	5	0.5	2,034.44	2,034.44	2	0.00	2,034.44	0.00
abs5n5.dat	5	0.5	2,362.16	2,362.16	6	0.00	2,413.72	2.18
abs1n10.dat	10	0.5	4,970.62	4,970.62	15	0.00	5,015.82	0.91
abs2n10.dat	10	0.5	4,803.17	4,803.17	13	0.00	5,142.58	7.07
abs3n10.dat	10	0.5	4,289.84	4,289.84	13	0.00	4,289.84	0.00
abs4n10.dat	10	0.5	4,347.06	4,347.06	11	0.00	4,347.06	0.00
abs5n10.dat	10	0.5	5,041.62	5,044.44	14	0.06	5,078.05	0.72
abs1n15.dat	15	0.5	5,713.84	5,713.84	37	0.00	5,713.84	0.00
abs2n15.dat	15	0.5	5,821.04	5,822.88	41	0.03	5,822.88	0.03
abs3n15.dat	15	0.5	6,711.25	6,711.25	45	0.00	6,894.43	2.73
abs4n15.dat	15	0.5	5,227.56	5,227.56	49	0.00	5,437.22	4.01
abs5n15.dat	15	0.5	5,210.85	5,215.02	61	0.08	5,424.97	4.11
abs1n20.dat	20	0.5	7,353.82	7,353.82	108	0.00	7,449.47	1.30
abs2n20.dat	20	0.5	7,385.03	7,385.03	101	0.00	7,450.67	0.89
abs3n20.dat	20	0.5	7,903.97	7,907.40	96	0.04	7,974.47	0.89
abs4n20.dat	20	0.5	7,050.91	7,050.91	143	0.00	7,601.67	7.81
abs5n20.dat	20	0.5	8,405.83	8,405.83	73	0.00	8,876.11	5.59
abs1n25.dat	25	0.5	8,657.70	8,657.70	205	0.00	8,985.82	3.79
abs2n25.dat	25	0.5	9,266.87	9,266.87	164	0.00	9,408.18	1.52
abs3n25.dat	25	0.5	9,843.60	9,843.60	208	0.00	9,843.60	0.00
abs4n25.dat	25	0.5	8,677.86	8,677.86	342	0.00	8,677.86	0.00
abs5n25.dat	25	0.5	10,857.68	10,857.68	191	0.00	10,857.68	0.00
abs1n30.dat	30	0.5	12,635.55	12,635.55	345	0.00	12,847.79	1.68
abs2n30.dat	30	0.5	11,351.36	11,351.36	273	0.00	11,487.63	1.20
abs3n30.dat	30	0.5	12,509.26	12,613.46	621	0.83	12,679.26	1.36
abs4n30.dat	30	0.5	9,928.35	9,928.35	573	0.00	10,018.88	0.91
abs5n30.dat	30	0.5	10,178.63	10,181.69	346	0.03	10,270.81	0.91
abs1n35.dat	35	0.5	11,984.69	11,984.69	600	0.00	12,382.03	3.32
abs2n35.dat	35	0.5	10,706.91	10,706.91	1,160	0.00	10,998.26	2.72
abs3n35.dat	35	0.5	14,411.58	14,478.78	581	0.47	14,732.69	2.23
abs4n35.dat	35	0.5	10,844.98	10,844.98	1,202	0.00	11,001.99	1.45
abs5n35.dat	35	0.5	11,195.87	11,195.87	624	0.00	11,365.71	1.52
abs1n40.dat	40	0.5	14,006.57	14,006.57	931	0.00	14,455.90	3.21
abs2n40.dat	40	0.5	11,722.58	11,722.58	996	0.00	11,941.41	1.87
abs3n40.dat	40	0.5	14,107.14	14,107.14	2,035	0.00	14,655.81	3.89
abs4n40.dat	40	0.5	11,684.43	11,684.43	1,276	0.00	11,763.03	0.67
abs5n40.dat	40	0.5	13,536.57	13,536.57	1,230	0.00	13,759.25	1.65
abs1n45.dat	45	0.5	14,661.20	14,661.20	2,099	0.00	14,867.01	1.40
abs2n45.dat	45	0.5	13,675.96	13,675.96	1,235	0.00	14,161.64	3.55
abs3n45.dat	45	0.5	15,316.57	15,316.57	1,416	0.00	15,564.91	1.62
abs4n45.dat	45	0.5	14,096.84	14,121.05	1,175	0.17	14,537.57	3.13
abs5n45.dat	45	0.5	13,838.54	13,840.06	1,747	0.01	14,215.84	2.73
abs1n50.dat	50	1	15,235.83	15,235.83	3,989	0.00	15,543.49	2.02
abs2n50.dat	50	1	15,453.78	15,455.34	2,412	0.01	15,630.97	1.15
abs3n50.dat	50	1	15,747.73	15,867.45	2,996	0.76	16,055.08	1.95
abs4n50.dat	50	1	17,163.52	17,173.36	2,169	0.06	17,450.53	1.67
abs5n50.dat	50	1	16,143.06	16,143.06	2,585	0.00	16,313.73	1.06

tions in a shorter amount of time than the other variants. In Tables 11 and 12, the different versions are run for the same number of iterations, and it clearly appears that the tabu search without any improvement procedure requires an amount of time similar to the time required by HAIR. This is because if no improvement procedure is applied, then there are many small improvements in the tabu search, and as a consequence, the Lin and Kernighan heuristic is run several times. On the contrary, HAIR improves s_{best} a smaller number of times by running the MILP models, each improvement being however larger. From Tables 9–12, we can also observe that the most significant improvement is due to MIP_2.

We also tested HAIR on larger instances where the comparison with the exact solution cannot be made because the exact approach is not able to solve them within a reasonable time. To speed up the solution process on these larger instances, some modifications were made to the algorithm for both policies.

1. The improvement procedure is called only if at least 20 iterations were performed since its last application.

2. The swap move is not considered.

Two classes of large instances were tested: the first with $h_i \in [0.01, 0.05]$ and $h_0 = 0.03$ and the second with $h_i \in [0.1, 0.5]$ and $h_0 = 0.03$. For both classes, the size of the instances is fixed to $n = 50, 100, 200$, with 10 instances for each size. The horizon is $H = 6$ and the remaining characteristics are the same as in the previous instances. Thus, for each class we have 30 instances. The results are shown in Tables 13 and 14. For each table, the first two columns report the name and the size of the instance. Then, the following seven columns refer to the OU policy: the first reports the value of parameter λ for the tabu list used in HAIR. The second column reports the best solution found by HAIR after one hour. The following four columns provide the error, with respect to the best solution, generated by HAIR when it is run for 5 minutes, 10 minutes, 30 minutes, and 1 hour, respectively. The following column gives the error of the solution generated by the heuristic presented in Bertazzi et al. (2002). The CPU time of this latter algorithm is very small: a few seconds on instances with up to 100 customers and always less than three minutes on instances with 200 customers. The last five columns refer to the ML policy, for which no known algorithm is available. The remaining columns have the same meaning as for the OU policy.

Finally, we tested HAIR on two sets of instances with $n = 200$ and time-varying demands. Only the demands change with respect to the previous set. The demands are generated as random numbers in the interval $[0.7\hat{r}_i, 1.3\hat{r}_i]$, where $\hat{r}_i$ is the demand of customer i in the previous set of tests. Moreover, $r_{0t} = 2\hat{r}_0$ and $C = 2\max_{t\in\mathcal{T}} \sum_{i\in\mathcal{M}} r_{it}$. In the first set, $h_i \in [0.01, 0.05]$ and $h_0 = 0.03$, and in the second $h_i \in [0.1, 0.5]$ and $h_0 = 0.03$. The results are shown in Tables 15 and 16. They cannot be compared with the heuristic proposed in Bertazzi et al. (2002) for the OU policy because this latter heuristic works only for time-constant demand. The results show that, for both policies, HAIR is able to find good solutions very quickly; even after five minutes, the error with respect to the best solution is always below 1%.

Table 7 OU Policy—Computational Results with $H = 6$ and $h_i \in [0.01, 0.05]$

Instance	n	λ	z^*	HAIR	CPU	% error	BPS	% error
abs1n5.dat	5	0.5	3,335.24	3,335.24	16	0.00	3,348.24	0.39
abs2n5.dat	5	0.5	2,722.33	2,722.33	23	0.00	2,722.33	0.00
abs3n5.dat	5	0.5	4,776.00	4,776.00	17	0.00	4,783.42	0.16
abs4n5.dat	5	0.5	3,246.66	3,246.66	21	0.00	3,389.66	4.40
abs5n5.dat	5	0.5	2,419.67	2,419.67	12	0.00	2,498.52	3.26
abs1n10.dat	10	0.5	4,499.25	4,499.25	78	0.00	4,534.62	0.79
abs2n10.dat	10	0.5	5,236.98	5,236.98	105	0.00	5,236.98	0.00
abs3n10.dat	10	0.5	4,652.53	4,652.53	67	0.00	4,652.53	0.00
abs4n10.dat	10	0.5	5,104.91	5,104.91	73	0.00	5,314.16	4.10
abs5n10.dat	10	0.5	4,670.76	4,670.76	61	0.00	4,760.99	1.93
abs1n15.dat	15	0.5	5,462.68	5,462.68	415	0.00	5,600.34	2.52
abs2n15.dat	15	0.5	5,494.74	5,494.74	360	0.00	5,907.40	7.51
abs3n15.dat	15	0.5	6,060.38	6,060.38	276	0.00	6,168.68	1.79
abs4n15.dat	15	0.5	5,504.65	5,504.65	380	0.00	6,027.52	9.50
abs5n15.dat	15	0.5	5,309.48	5,309.48	256	0.00	5,311.46	0.04
abs1n20.dat	20	2	6,490.18	6,501.26	559	0.17	6,541.11	0.78
abs2n20.dat	20	2	6,082.54	6,093.54	987	0.18	6,348.07	4.37
abs3n20.dat	20	2	6,950.20	6,953.78	733	0.05	7,186.46	3.40
abs4n20.dat	20	2	7,432.78	7,432.78	698	0.00	7,729.58	3.99
abs5n20.dat	20	2	7,210.73	7,210.73	1,212	0.00	7,369.90	2.21
abs1n25.dat	25	2	7,095.86	7,095.86	1,495	0.00	7,595.85	7.05
abs2n25.dat	25	2	7,484.84	7,504.84	1,188	0.27	8,067.75	7.79
abs3n25.dat	25	2	7,728.76	7,728.76	3,170	0.00	7,996.62	3.47
abs4n25.dat	25	2	7,509.02	7,509.02	1,251	0.00	8,035.38	7.01
abs5n25.dat	25	2	7,452.28	7,518.61	1,496	0.89	7,871.75	5.63
abs1n30.dat	30	2	8,319.59	8,349.59	2,334	0.36	8,761.13	5.31
abs2n30.dat	30	2	7,761.53	7,768.53	3,251	0.09	8,049.36	3.71
abs3n30.dat	30	2	8,214.55	8,365.82	3,654	1.84	8,654.55	5.36
abs4n30.dat	30	2	7,574.80	7,574.80	5,146	0.00	8,067.12	6.50
abs5n30.dat	30	2	7,366.47	7,402.71	2,220	0.49	7,538.91	2.34

Table 8 OU Policy—Computational Results with $H = 6$ and $h_i \in [0.1, 0.5]$

Instance	n	λ	z^*	HAIR	CPU	% error	*BPS*	% error
abs1n5.dat	5	0.5	5,942.82	5,942.82	17	0.00	5,942.82	0.00
abs2n5.dat	5	0.5	5,045.91	5,045.91	20	0.00	5,047.79	0.04
abs3n5.dat	5	0.5	6,956.28	6,956.28	16	0.00	6,969.76	0.19
abs4n5.dat	5	0.5	5,163.42	5,163.42	29	0.00	5,226.16	1.22
abs5n5.dat	5	0.5	4,581.66	4,581.66	17	0.00	4,593.05	0.25
abs1n10.dat	10	0.5	8,870.15	8,870.15	78	0.00	8,945.11	0.85
abs2n10.dat	10	0.5	8,569.73	8,569.73	66	0.00	8,999.23	5.01
abs3n10.dat	10	0.5	8,509.81	8,509.81	129	0.00	8,509.81	0.00
abs4n10.dat	10	0.5	8,792.29	8,792.29	112	0.00	8,994.79	2.30
abs5n10.dat	10	0.5	9,620.07	9,620.07	67	0.00	9,735.40	1.20
abs1n15.dat	15	0.5	12,118.83	12,118.83	266	0.00	12,118.83	0.00
abs2n15.dat	15	0.5	11,932.10	11,932.1	305	0.00	12,158.12	1.89
abs3n15.dat	15	0.5	13,554.15	13,554.15	369	0.00	13,554.15	0.00
abs4n15.dat	15	0.5	10,618.55	10,618.55	285	0.00	10,711.77	0.88
abs5n15.dat	15	0.5	10,385.54	10,385.54	220	0.00	10,721.92	3.24
abs1n20.dat	20	2	14,702.95	14,722.95	507	0.14	15,250.81	3.73
abs2n20.dat	20	2	14,646.96	14,670.34	746	0.16	14,785.04	0.94
abs3n20.dat	20	2	14,532.91	14,577.14	863	0.30	14,764.08	1.59
abs4n20.dat	20	2	14,539.72	14,539.72	1,084	0.00	14,590.96	0.35
abs5n20.dat	20	2	15,896.71	15,904	533	0.04	16,506.45	3.84
abs1n25.dat	25	2	15,581.47	15,610.98	1,941	0.19	15,726.50	0.93
abs2n25.dat	25	2	16,823.16	16,843.16	1,544	0.12	17,017.02	1.15
abs3n25.dat	25	2	18,098.02	18,121.26	2,198	0.13	18,506.33	2.26
abs4n25.dat	25	2	16,303.69	16,303.69	1,368	0.00	17,026.13	4.43
abs5n25.dat	25	2	19,047.70	19,080.27	1,856	0.17	19,394.10	1.82
abs1n30.dat	30	2	23,183.99	23,183.99	2,365	0.00	23,754.83	2.46
abs2n30.dat	30	2	20,090.29	20,159.96	2,178	0.35	20,713.82	3.10
abs3n30.dat	30	2	23,382.73	23,439.91	3,978	0.24	23,918.93	2.29
abs4n30.dat	30	2	17,649.53	17,746.79	5,063	0.55	18,247.06	3.39
abs5n30.dat	30	2	18,979.93	18,997.61	2,239	0.09	19,270.91	1.53

Table 9 ML Policy—Impact of the Improvement Phase with $H = 6$ and $h_i \in [0.1, 0.5]$

	HAIR		Tabu		Tabu + 1		Tabu + 2		Tabu + 3		Tabu + 1 + 2		Tabu + 1 + 3		Tabu + 2 + 3	
Instance	CPU	% error	CPU	% error	CPU	% error	CPU	% error	CPU	% error	CPU	% error	CPU	% error	CPU	% error
abs1n30.dat	1,922	0.16	2,506	1.73	1,500	2.51	1,870	0.17	1,326	2.50	2,111	0.17	1,338	2.50	1,862	0.16
abs2n30.dat	1,850	0.61	2,125	0.75	2,114	0.29	2,107	1.00	1,848	0.32	2,968	0.93	2,698	0.06	3,654	0.43
abs3n30.dat	2,192	0.44	1,162	1.47	1,193	1.49	1,172	1.49	1,196	1.46	1,186	1.49	1,184	1.47	1,223	1.46
abs4n30.dat	1,952	0.23	3,571	0.79	5,292	1.19	1,991	0.30	3,582	0.78	2,030	0.30	3,632	0.78	1,966	0.23
abs5n30.dat	1,694	0.21	4,428	0.89	2,448	0.44	1,787	0.21	2,663	0.28	1,809	0.21	3,259	0.05	1,806	0.21

Table 10 OU Policy—Impact of the Improvement Phase with $H = 6$ and $h_i \in [0.1, 0.5]$

	HAIR		Tabu		Tabu + 1		Tabu + 2		Tabu + 3		Tabu + 1 + 2		Tabu + 1 + 3		Tabu + 2 + 3	
Instance	CPU	% error	CPU	% error	CPU	% error	CPU	% error	CPU	% error	CPU	% error	CPU	% error	CPU	% error
abs1n30.dat	2,365	0.00	8,785	2.45	3,648	3.09	1,831	0.96	4,188	2.36	2,913	0.74	4,992	2.36	2,421	0.74
abs2n30.dat	2,178	0.35	6,612	0.00	2,872	0.00	1,995	0.03	2,790	0.00	4,562	0.00	2,250	0.00	2,361	0.00
abs3n30.dat	3,978	0.24	2,701	1.07	6,365	0.09	3,460	0.09	3,392	0.99	3,476	0.61	3,550	0.98	3,071	0.67
abs4n30.dat	5,063	0.55	6,352	0.21	3,105	1.71	3,129	0.43	4,608	1.74	3,191	0.43	4,340	1.61	2,760	0.91
abs5n30.dat	2,239	0.09	4,556	1.12	4,643	0.03	2,263	0.03	4,663	1.12	2,981	0.13	2,374	1.29	3,095	0.03

Table 11 ML Policy for 10,000 Iterations—Impact of the Improvement Phase with $H = 6$ and $h_i \in [0.1, 0.5]$

	HAIR		Tabu		Tabu + 1		Tabu + 2		Tabu + 3		Tabu + 1 + 2		Tabu + 1 + 3		Tabu + 2 + 3	
Instance	CPU	% error	CPU	% error	CPU	% error	CPU	% error	CPU	% error	CPU	% error	CPU	% error	CPU	% error
abs1n30.dat	605	0.16	225	2.53	271	2.51	573	0.20	223	2.50	592	0.16	224	2.50	581	0.16
abs2n30.dat	516	0.87	509	0.46	439	0.86	529	0.85	518	0.08	533	0.62	533	0.08	486	0.87
abs3n30.dat	227	1.49	285	1.55	248	1.55	258	1.55	209	1.49	253	1.55	212	1.49	208	1.49
abs4n30.dat	601	0.14	629	1.75	553	1.73	561	0.96	613	1.68	546	0.96	614	1.68	571	0.14
abs5n30.dat	525	0.12	566	1.74	598	1.71	575	0.07	523	1.64	539	0.53	523	1.70	493	0.12

Table 12 OU Policy for 10,000 Iterations—Impact of the Improvement Phase with $H = 6$ and $h_i \in [0.1, 0.5]$

	HAIR		Tabu		Tabu + 1		Tabu + 2		Tabu + 3		Tabu + 1 + 2		Tabu + 1 + 3		Tabu + 2 + 3	
Instance	CPU	% error	CPU	% error	CPU	% error	CPU	% error	CPU	% error	CPU	% error	CPU	% error	CPU	% error
abs1n30.dat	817	0.01	531	1.79	592	1.79	755	1.05	591	1.79	847	1.05	572	1.79	708	0.01
abs2n30.dat	664	0.00	565	0.54	724	0.22	581	0.30	616	0.00	695	0.19	634	0.31	631	0.00
abs3n30.dat	760	0.68	766	1.13	778	0.63	742	0.31	740	0.68	778	0.31	787	0.31	723	0.68
abs4n30.dat	765	0.43	792	0.61	842	0.43	716	0.43	738	0.11	752	0.43	898	2.24	732	0.43
abs5n30.dat	679	0.13	703	1.29	723	1.70	662	0.13	731	1.73	691	0.13	729	1.29	664	0.13

Table 13 Computational Results on Larger Instances with $h_i \in [0.1, 0.5]$ and $H = 6$

		OU policy							ML policy				
Instance	n	λ	z_{best}	HAIR (5 min)	HAIR (10 min)	HAIR (30 min)	HAIR (1 hour)	BPS	λ	z_{best}	HAIR (1 min)	HAIR (5 min)	HAIR (30 min)
abs1n50.dat	50	4	31,147.82	0.00	0.00	0.00	0.00	1.55	0.5	30,225.36	0.00	0.00	0.00
abs2n50.dat	50	4	30,192.51	0.17	0.17	0.00	0.00	1.85	0.5	29,856.26	0.00	0.00	0.00
abs3n50.dat	50	4	30,420.96	0.00	0.00	0.00	0.00	2.17	0.5	29,904.15	0.00	0.00	0.00
abs4n50.dat	50	4	31,898.84	0.91	0.00	0.00	0.00	2.19	0.5	31,677.87	0.18	0.00	0.00
abs5n50.dat	50	4	29,518.68	0.83	0.00	0.00	0.00	0.56	0.5	29,400.33	0.00	0.00	0.00
abs6n50.dat	50	4	32,394.50	1.50	0.00	0.00	0.00	1.42	0.5	31,946.33	0.78	0.78	0.78
abs7n50.dat	50	4	30,165.00	1.52	1.52	0.29	0.00	3.06	0.5	29,768.03	0.00	0.00	0.00
abs8n50.dat	50	4	26,416.46	1.20	1.20	0.05	0.05	0.00	0.5	26,521.96	0.00	0.00	0.00
abs9n50.dat	50	4	30,671.88	1.52	0.00	0.00	0.00	1.94	0.5	30,283.9	0.00	0.00	0.00
abs10n50.dat	50	4	32,362.01	0.00	0.00	0.00	0.00	1.71	0.5	31,397.84	0.00	0.00	0.00

Table 13 (Continued)

		OU policy						ML policy				
Instance	n	λ	z_{best}	HAIR (5 min)	HAIR (10 min)	HAIR (30 min)	HAIR (1 hour)	BPS	λ	z_{best}	HAIR (1 min)	HAIR (5 min)	HAIR (30 min)
abs1n100.dat	100	4	57,962.58	0.34	0.33	0.00	0.00	2.58	0.5	57,721.23	0.03	0.00	0.00
abs2n100.dat	100	4	53,942.07	1.58	1.58	0.82	0.00	2.72	0.5	53,432.8	0.00	0.00	0.00
abs3n100.dat	100	4	59,370.69	1.72	1.37	1.36	0.00	0.70	0.5	58,598.93	0.08	0.00	0.00
abs4n100.dat	100	4	52,419.33	1.58	1.37	0.00	0.00	1.68	0.5	52,030.59	0.00	0.00	0.00
abs5n100.dat	100	4	58,679.72	1.54	1.40	0.47	0.00	0.59	0.5	58,258.92	0.32	0.00	0.00
abs6n100.dat	100	4	55,756.46	1.58	1.56	1.56	0.00	1.88	0.5	55,280.01	0.37	0.34	0.00
abs7n100.dat	100	4	56,861.23	0.16	0.03	0.00	0.00	1.64	0.5	56,398.19	0.00	0.00	0.00
abs8n100.dat	100	4	55,741.08	1.06	1.05	0.42	0.00	2.33	0.5	55,384.47	0.61	0.61	0.61
abs9n100.dat	100	4	59,219.73	1.63	1.52	1.08	0.00	2.45	0.5	58,729.94	0.00	0.00	0.00
abs10n100.dat	100	4	57,094.63	0.51	0.51	0.00	0.00	2.10	0.5	56,644.22	0.38	0.00	0.00
abs1n200.dat	200	4	112,090.80	0.46	0.46	0.08	0.00	0.47	0.5	110,790.39	0.02	0.01	0.00
abs2n200.dat	200	4	114,647.69	0.03	0.03	0.00	0.00	0.55	0.5	112,401.98	0.34	0.34	0.04
abs3n200.dat	200	4	109,339.50	0.68	0.68	0.68	0.00	1.01	0.5	108,119.61	0.00	0.00	0.00
abs4n200.dat	200	4	110,841.13	0.10	0.10	0.02	0.00	0.36	0.5	109,309.48	0.06	0.06	0.00
abs5n200.dat	200	4	110,994.17	0.00	0.00	0.00	0.00	0.38	0.5	109,083.07	0.11	0.11	0.02
abs6n200.dat	200	4	110,433.85	0.26	0.26	0.00	0.00	0.93	0.5	109,071.2	0.16	0.01	0.00
abs7n200.dat	200	4	98,880.57	0.71	0.71	0.15	0.00	1.06	0.5	97,749.52	0.08	0.08	0.00
abs8n200.dat	200	4	104,334.63	0.08	0.08	0.05	0.00	0.10	0.5	102,194.63	0.10	0.02	0.00
abs9n200.dat	200	4	106,994.52	0.18	0.18	0.08	0.00	0.14	0.5	104,877.49	0.22	0.22	0.00
abs10n200.dat	200	4	110,540.23	0.44	0.44	0.32	0.00	0.63	0.5	109,045.17	0.19	0.08	0.02

Table 14 Computational Results on Larger Instances with $h_i \in [0.01, 0.05]$ and $H = 6$

		OU policy						ML policy				
Instance	n	λ	z_{best}	HAIR (5 min)	HAIR (10 min)	HAIR (30 min)	HAIR (1 hour)	BPS	λ	z_{best}	HAIR (5 min)	HAIR (10 min)	HAIR (30 min)
abs1n50.dat	50	4	10,409.13	4.44	4.44	0.00	0.00	5.74	0.5	9,974.38	0.00	0.00	0.00
abs2n50.dat	50	4	10,881.35	5.32	0.00	0.00	0.00	5.31	0.5	10,632.24	0.58	0.00	0.00
abs3n50.dat	50	4	10,767.39	0.00	0.00	0.00	0.00	6.63	0.5	10,548.98	0.00	0.00	0.00
abs4n50.dat	50	4	10,656.21	4.88	4.88	0.00	0.00	10.30	0.5	10,555.38	0.00	0.00	0.00
abs5n50.dat	50	4	10,234.60	1.57	1.57	0.00	0.00	8.57	0.5	10,137.4	0.00	0.00	0.00
abs6n50.dat	50	4	10,533.63	0.86	0.86	0.86	0.00	5.79	0.5	10,166.1	0.00	0.00	0.00
abs7n50.dat	50	4	10,460.82	3.67	3.67	3.29	2.37	0.00	0.5	10,012.68	0.01	0.00	0.00
abs8n50.dat	50	4	10,411.20	4.83	2.49	2.49	0.00	9.48	0.5	10,547.97	2.35	2.35	0.00
abs9n50.dat	50	4	10,305.69	1.63	1.63	1.63	0.00	2.49	0.5	10,052.78	0.87	0.87	0.00
abs10n50.dat	50	4	10,470.63	0.00	0.00	0.00	0.00	1.00	0.5	9,727.74	0.00	0.00	0.00
abs1n100.dat	100	4	15,938.80	0.44	0.03	0.03	0.00	4.38	0.5	15,784.02	0.07	0.00	0.00
abs2n100.dat	100	4	14,920.94	5.61	5.61	0.00	0.00	3.01	0.5	14,754.1	0.00	0.00	0.00
abs3n100.dat	100	4	15,836.71	7.44	6.07	5.82	0.00	8.35	0.5	15,649.44	0.38	0.00	0.00
abs4n100.dat	100	4	15,097.32	1.42	0.67	0.00	0.00	4.39	0.5	14,755.76	0.34	0.00	0.00
abs5n100.dat	100	4	15,697.01	4.84	4.84	0.00	0.00	1.49	0.5	15,412.83	0.48	0.00	0.00
abs6n100.dat	100	4	15,549.57	5.62	5.46	5.09	0.00	3.38	0.5	15,327.08	0.95	0.58	0.00
abs7n100.dat	100	4	15,725.24	0.54	0.00	0.00	0.00	3.59	0.5	15,468.98	0.66	0.58	0.00
abs8n100.dat	100	4	15,280.84	4.29	4.29	4.29	0.00	6.26	0.5	15,078.16	1.95	0.22	0.00
abs9n100.dat	100	4	16,471.32	0.06	0.01	0.00	0.00	2.69	0.5	15,628.66	0.00	0.00	0.00
abs10n100.dat	100	4	15,618.39	4.27	3.55	1.72	0.00	4.90	0.5	15,495.87	0.07	0.00	0.00
abs1n200.dat	200	4	25,690.21	0.01	0.01	0.01	0.00	1.06	0.5	24,353.4	0.57	0.57	0.00
abs2n200.dat	200	4	25,252.21	3.89	3.89	3.89	0.00	4.93	0.5	24,576.55	0.45	0.45	0.14
abs3n200.dat	200	4	25,170.57	1.07	1.07	0.43	0.43	0.00	0.5	23,861.65	0.81	0.81	0.00
abs4n200.dat	200	4	24,607.65	3.82	3.82	3.82	0.00	4.76	0.5	24,232.73	1.41	1.41	0.40
abs5n200.dat	200	4	25,263.32	1.57	1.57	1.55	1.40	0.00	0.5	24,193.98	0.72	0.72	0.00
abs6n200.dat	200	4	24,213.10	0.17	0.17	0.04	0.00	5.72	0.5	23,651.81	0.95	0.95	0.00
abs7n200.dat	200	4	24,731.63	0.02	0.02	0.00	0.00	1.02	0.5	23,527.12	0.47	0.47	0.00
abs8n200.dat	200	4	23,617.79	1.29	1.29	0.35	0.00	5.79	0.5	23,230.42	0.81	0.81	0.35
abs9n200.dat	200	4	25,387.67	0.39	0.39	0.35	0.00	0.14	0.5	23,846.19	1.20	1.20	0.00
abs10n200.dat	200	4	23,937.25	2.57	1.12	0.61	0.00	2.54	0.5	23,609.34	0.61	0.61	0.00

Table 15 Computational Results on Instances with $n = 200$, Time-Varying Demand, $h_i \in [0.1, 0.5]$, and $H = 6$

		OU policy					ML policy			
Instance	n	λ	z_{best}	HAIR (5 min)	HAIR (10 min)	HAIR (30 min)	λ	z_{best}	HAIR (5 min)	HAIR (10 min)	HAIR (30 min)
abs1n200.dat	200	4	118,367.67	0.74	0.16	0.02	0.5	115,735.09	0.62	0.27	0.00
abs2n200.dat	200	4	120,248.85	0.20	0.20	0.04	0.5	117,885.07	0.25	0.03	0.00
abs3n200.dat	200	4	121,552.35	0.15	0.15	0.04	0.5	119,282.59	0.48	0.02	0.00
abs4n200.dat	200	4	116,787.99	0.25	0.25	0.15	0.5	114,408.63	0.11	0.10	0.03
abs5n200.dat	200	4	117,670.2	0.09	0.07	0.00	0.5	115,964.55	0.49	0.05	0.00
abs6n200.dat	200	4	117,593.32	0.10	0.10	0.06	0.5	115,268.77	0.40	0.09	0.00
abs7n200.dat	200	4	117,952.38	0.17	0.09	0.02	0.5	115,035.23	0.32	0.02	0.00
abs8n200.dat	200	4	106,770.74	0.22	0.22	0.10	0.5	104,863.2	0.28	0.01	0.00
abs9n200.dat	200	4	111,369.2	0.23	0.23	0.08	0.5	108,743.66	0.16	0.03	0.00
abs10n200.dat	200	4	182,039.55	0.13	0.11	0.06	0.5	179,767.5	0.19	0.02	0.00

Table 16 Computational Results on Instances with $n = 200$, Time-Varying Demand, $h_i \in [0.01, 0.05]$, and $H = 6$

		OU policy					ML policy			
Instance	n	λ	z_{best}	HAIR (5 min)	HAIR (10 min)	HAIR (30 min)	λ	z_{best}	HAIR (5 min)	HAIR (10 min)	HAIR (30 min)
abs1n200.dat	200	4	31,287.17	0.63	0.28	0.14	0.5	30,874.83	0.39	0.13	0.00
abs2n200.dat	200	4	31,974.67	0.57	0.57	0.31	0.5	31,499.76	0.81	0.10	0.00
abs3n200.dat	200	4	32,294.22	0.79	0.00	0.00	0.5	32,022.87	0.16	0.01	0.00
abs4n200.dat	200	4	31,121.91	0.36	0.36	0.18	0.5	30,739.6	0.21	0.00	0.00
abs5n200.dat	200	4	31,701.76	0.91	0.67	0.31	0.5	31,569.27	1.25	0.54	0.38
abs6n200.dat	200	4	31,605.54	0.45	0.45	0.29	0.5	31,139.25	0.98	0.64	0.03
abs7n200.dat	200	4	31,178	0.13	0.10	0.06	0.5	30,516.92	0.88	0.03	0.00
abs8n200.dat	200	4	31,117.86	0.35	0.16	0.16	0.5	30,748.9	0.91	0.06	0.00
abs9n200.dat	200	4	31,336.07	0.19	0.19	0.18	0.5	30,683.35	0.44	0.10	0.00
abs10n200.dat	200	4	37,947.81	1.43	1.12	0.25	0.5	37,857.39	0.43	0.23	0.00

5. Conclusions

In this paper we presented a hybrid heuristic, HAIR, that combines a tabu search scheme with an improvement phase, applied whenever the best incumbent solution is updated, that implies the run of MIP models. Such MIPs explore in depth the neighborhood of the incumbent solution. Although the MIP models are shown to be NP-hard, their exact solution did not create any computational problem on the tested instances, that is up to 200 customers with a horizon of six time units. On larger size instances, a time limit may be set for the solution of a MIP, and the best solution obtained within that time limit may be used instead of the optimal solution. The availability of the optimal solutions allowed us to calculate the exact errors generated by HAIR on small size instances. The average error is 0.08% for the OU policy and 0.05% for the ML policy. On instances with up to 200 customers and a horizon of $H = 6$ time units, HAIR substantially improves the solutions obtained by a known heuristic for the case of the OU policy.

Electronic Companion

An electronic companion to this paper is available as part of the online version that can be found at http://joc.journal.informs.org/.

Acknowledgments

The authors acknowledge the comments and suggestions of two anonymous reviewers that have helped them improve a previous version of this paper.

References

Archetti, C., L. Bertazzi, G. Laporte, M. G. Speranza. 2007. A branch-and-cut algorithm for a vendor-managed inventory routing problem. *Transportation Sci.* **41**(3) 382–391.

Bell, W. J., L. M. Dalberto, M. L. Fisher, A. J. Greenfield, R. G. Jaikumar, P. Kedia, R. G. Mack, P. J. Prutzman. 1983. Improving the distribution of industrial gases with an on-line computerized routing and scheduling optimizer. *Interfaces* **13**(6) 4–23.

Bertazzi, L., G. Paletta, M. G. Speranza. 2002. Deterministic order-up-to level policies in an inventory routing problem. *Transportation Sci.* **36**(1) 119–132.

Bertazzi, L., M. G. Speranza, M. W. P. Savelsbergh. 2008. Inventory routing. B. Golden, R. Raghavan, E. Wasil, eds. *The Vehicle Routing Problem: Latest Advances and New Challenges*. Springer, New York, 49–72.

Blumenfeld, D. E., L. D. Burns, J. D. Diltz, C. F. Daganzo. 1985. Analyzing trade-offs between transportation, inventory and production costs on freight networks. *Transportation Res. B* **19**(5) 361–380.

Campbell, A. M., L. Clarke, A. Kleywegt, M. W. P. Savelsbergh. 1998. The inventory routing problem. T. G. Crainic, G. Laporte, eds. *Fleet Management and Logistics*. Kluwer Academic Publishers, Boston, 95–113.

Cordeau, J.-F., G. Laporte, M. W. P. Savelsbergh, D. Vigo. 2009. Vehicle routing. C. Barnhart, G. Laporte, eds. *Handbooks in Operations Research and Management Science: Transportation*, Vol. 14. North-Holland, Amsterdam, 367–428.

Dror, M., M. Ball, B. Golden. 1985. A computational comparison of algorithms for the inventory routing problem. *Ann. Oper. Res.* **4**(1) 1–23.

Federgruen, A., D. Simchi-Levi. 1995. Analysis of vehicle routing and inventory-routing problems. M. O. Ball, T. L. Magnanti, C. L. Monma, G. L. Nemhauser, eds. *Handbooks in Operations Research and Management Science: Network Routing*, Vol. 8. North-Holland, Amsterdam, 297–373.

Federgruen, A., P. Zipkin. 1984. A combined vehicle routing and inventory allocation problem. *Oper. Res.* **32**(5) 1019–1032.

Garey, M. R., D. S. Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Company, New York.

Gendreau, M., A. Hertz, G. Laporte. 1994. A tabu search heuristic for the vehicle routing problem. *Management Sci.* **40**(10) 1276–1290.

Glover, F. 1986. Future paths for integer programming and links to artificial intelligence. *Comput. Oper. Res.* **13**(5) 533–549.

Glover, F., M. Laguna. 1997. *Tabu Search*. Kluwer Academic Publishers, Dordrecht, The Netherlands.

Golden, B., A. Assad, R. Dahl. 1984. Analysis of a large scale vehicle routing problem with an inventory component. *Large Scale Systems* **7**(2–3) 181–190.

Kellerer, H., U. Pferschy, D. Pisinger. 2004. *Knapsack Problems*. Springer-Verlag, Berlin.

Lin, S., B. W. Kernighan. 1973. An effective heuristic algorithm for the traveling-salesman problem. *Oper. Res.* **21**(2) 498–516.

Martello, S., P. Toth. 1990 *Knapsack Problems: Algorithms and Computer Implementations*. John Wiley & Sons, New York.

Taillard, E. 1991. Robust taboo search for the QAP. *Parallel Comput.* **17**(4–5) 443–455.