A Hybrid Heuristic for an Inventory-Routing Problem

Claudia Archetti [(1)] *Luca Bertazzi* [(1)] *Alain Hertz* [(2)]
M. Grazia Speranza [(1)]

[(1)] *University of Brescia, Department of Quantitative Methods, Brescia, Italy*
[(2)] *École Polytechnique and GERAD, Montréal, Canada*
{archetti, bertazzi, speranza}@eco.unibs.it
alain.hertz@gerad.ca

Abstract

We consider an inventory routing problem in discrete time where a supplier has to serve a set of customers over a time horizon. A capacity constraint for the inventory is given for each customer and the service cannot cause any stock-out 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.

Keywords: Inventory–Routing Problem, metaheuristic, tabu search, optimization, integer programming.

Introduction

The class of Inventory–Routing Problems (IRP) includes a variety of different optimization problems that, though 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, inventory holding costs may be accounted for in the objective function or not. 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, if compared to vehicle routing problems. However, the interest in this class of problems has been increasing from the beginning of the eighties. Some pioneering papers appeared in the eighties (see, e.g., [2, 5, 11, 13, 18]), while several papers appeared in the last two decades and some surveys (see, e.g., [4, 9, 10, 12]) 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. The only exact approach for an IRP we are aware of is presented in [1]. The availability of exact solutions for test instances makes it 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 proved to be very effective for many vehicle routing problems, the complexity of the IRP we studied required, in order to get high quality solutions, a more sophisticated heuristic search of the solution space. The effectiveness of the heuristic is proved over a set of benchmark instances.

The paper is organized as follows. In Section 1 we describe the IRP we study. In Section 2 we describe the hybrid heuristic we propose and in Section 3 we mention the computational experiments and show the obtained results.

1 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 H. 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 non–negative value that does not violate the capacity U_i. The

inventory holding cost is charged both at the supplier and at the customers. 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 in order 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 customer i to customer 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 $\sum_{i\in\mathcal{M}'} \sum_{j\in\mathcal{M}', i\neq j} \sum_{t\in\mathcal{T}} c_{ij} y_{ij}^t$.

To be *feasible*, a solution should not have any stock-out 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 be not 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.

An exact approach to this problem, with both replenishment policies, OU and ML, was proposed in [1]. A similar problem that incorporates production decisions, with ML policy and no inventory cost at the customers, was analyzed in [6], [7] and [8], where the authors proposed different heuristic algorithms to solve the problem.

2 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 which would bring the search back to a recently visited solution. Tabu search was introduced in [16]. A description of the method and its concepts can be found in [17].

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.

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 which 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; *Jump* performs changes on s in order to jump to a new region of the search space S. In the next sub-sections 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.

Algorithm 1 HAIR - A hybrid heuristic

Apply the *Initialization* procedure to generate an initial solution s. Set $s_{best} \leftarrow s$.
while the number of iterations without improvement of $s_{best} \leq MaxIter$ **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 $s_{best} \leftarrow s'$.
  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

2.1 Tabu search ingredients

As mentioned in Section 1, a solution to the IRP is feasible if there is no stock-out, 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 which have no stock-out at the customers, while stock-out 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

- $B_t(s)$ 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:

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

where $[.]^+ = max\{., 0\}$ and α and β are penalty factors initialized to 1 and updated every 10 iterations as in [15]: 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 while if they were all infeasible the value of α (β) is doubled.

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 stock-out 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. The length of both tabu lists is equal to

$$L + random(\lambda\sqrt{nR(s)}), \tag{1}$$

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 $\{0, \cdots, x\}$.

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 stock-out at customer i in order to keep the solution admissible.

- For the ML policy, if there is no stock-out at i when removing the visit, then nothing else is made; otherwise, the removal is performed only if the stock-out 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 stock-out 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 $I_{it'} + x_{it'} + (x - y) \leq U_i$, 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 stock-out 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 which 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, a time $t \in T_i(s)$ and a time $t' \notin T_i(s)$, 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 visit to i at time t and the visit to j at time t', and add a new visit to i at time t' and a new visit 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 stock-out 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 stock-out 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 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 stock-out (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 $\mathcal{A}$ while 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 the Algorithm 2 that can be found in the Appendix. The procedure *Move* chooses the best solution in $N(s)$ as next current solution.

2.2 Improvement Phase

We have designed an *Improvement* procedure which 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.

2.2.1 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 saving is calculated.

Let Δ_{ir} be the transportation saving gained if customer i is removed from route r (which, obviously, visits customer i in the incumbent solution). This saving 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 $MIP_1(OU, s)$, is formulated as follows:

$$\text{minimize} \quad \sum_{t\in\mathcal{T}'} h_0 B_t + \sum_{i\in\mathcal{M}}\sum_{t\in\mathcal{T}'} h_i I_{it} - \sum_{i\in\mathcal{M}}\sum_{r\in\mathcal{R}} \Delta_{ir} w_{ir} \tag{2}$$

subject to

$$B_t = B_{t-1} + r_{0t-1} - \sum_{i\in\mathcal{M}} x_{it-1} \qquad t \in \mathcal{T}' \tag{3}$$

$$B_t \geq \sum_{i\in\mathcal{M}} x_{it} \qquad t \in \mathcal{T} \tag{4}$$

$$I_{it} = I_{it-1} + x_{it-1} - r_{it-1} \qquad i \in \mathcal{M} \quad t \in \mathcal{T}' \tag{5}$$

$$x_{it} \geq U_i\theta_{it} - I_{it} \qquad i \in \mathcal{M} \quad t \in \mathcal{T} \tag{6}$$

$$x_{it} \leq U_i - I_{it} \qquad i \in \mathcal{M} \quad t \in \mathcal{T} \tag{7}$$

$$x_{it} \leq U_i\theta_{it} \qquad i \in \mathcal{M} \quad t \in \mathcal{T} \tag{8}$$

$$\sum_{i\in\mathcal{M}} x_{it} \leq C \qquad t \in \mathcal{T} \tag{9}$$

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

$$\sum_{r\in\mathcal{R}} z_{rt} \leq 1 \qquad t \in \mathcal{T} \tag{11}$$

$$x_{it} \leq U_i \sum_{r\in\mathcal{R}} (\sigma_{ir} z_{rt}) \qquad s \in \mathcal{M} \quad t \in \mathcal{T} \tag{12}$$

$$x_{it} \leq U_i[2 - \sigma_{ir}(z_{rt} + w_{ir})] \qquad i \in \mathcal{M} \quad r \in \mathcal{R} \quad t \in \mathcal{T} \tag{13}$$

$$w_{ir} \leq \sigma_{ir} \sum_{t\in\mathcal{T}} z_{rt} \qquad i \in \mathcal{M} \quad r \in \mathcal{R} \tag{14}$$

$$x_{it} \geq 0 \qquad i \in \mathcal{M} \quad t \in \mathcal{T} \tag{15}$$

$$I_{it} \geq 0 \qquad i \in \mathcal{M} \quad t \in \mathcal{T}' \tag{16}$$

$$B_t \geq 0 \qquad t \in \mathcal{T}' \tag{17}$$

$$\theta_{it} \in \{0, 1\} \qquad i \in \mathcal{M} \quad t \in \mathcal{T} \tag{18}$$

$$w_{ir} \in \{0, 1\} \qquad i \in \mathcal{M} \quad r \in \mathcal{R} \tag{19}$$

$$z_{rt} \in \{0, 1\} \qquad r \in \mathcal{R} \quad t \in \mathcal{T} \tag{20}$$

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 saving generated by the

customers that are removed from routes where they are visited in the incumbent solution. Constraints (3)-(5) (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 stock-out at the supplier. Constraints (6)-(8) define the OU policy and (9) is the vehicle capacity constraint. Constraints (10) establish that a route r can be assigned to at most one time period, while constraints (11) impose a maximum of one route at each time $t \in \mathcal{T}$. Constraints (12) impose that a customer can be served at time t only if it is visited by the route assigned at time t. Constraints (13) 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 (14) 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 saving component in the objective function is only an estimation of the real saving generated by the removal of the customers.

We do not consider the possibility of replicating a single route on different days (constraint (10)) because this created a remarkable increase in the complexity of the model and made the problem unsolvable in reasonable time. The same 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 saving gained if customer i is removed from the route assigned to time t (which, obviously, visits customer i in the incumbent solution). This saving 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 v_{it} be a binary variable equal to 1 if customer i is inserted into the route at time t. Note that when compared to the previous MIP, the index t replaces the index r in Δ_{it}, σ_{it} and w_{it}, the reason being that the routes are now assigned to specific times. Moreover, the v_{it} are new variables needed since 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 $MIP_2(OU, s)$. 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:

$$\text{minimize} \quad \sum_{t \in \mathcal{T}'} h_0 B_t + \sum_{i \in \mathcal{M}} \sum_{t \in \mathcal{T}'} h_i I_{it} - \sum_{i \in \mathcal{M}} \sum_{t \in \mathcal{T}} \Delta_{it} w_{it} + \sum_{i \in \mathcal{M}} \sum_{t \in \mathcal{T}} \Gamma_{it} v_{it} \tag{21}$$

subject to constraints (3)-(9), (15)-(18) and to

$$v_{it} \leq 1 - \sigma_{it} \qquad i \in \mathcal{M} \quad t \in \mathcal{T} \tag{22}$$
$$w_{it} \leq \sigma_{it} \qquad i \in \mathcal{M} \quad t \in \mathcal{T} \tag{23}$$
$$x_{it} \leq U_i(\sigma_{it} - w_{it} + v_{it}) \qquad i \in \mathcal{M} \quad t \in \mathcal{T} \tag{24}$$
$$v_{it} \in \{0, 1\} \qquad i \in \mathcal{M} \quad t \in \mathcal{T} \tag{25}$$
$$w_{it} \in \{0, 1\} \qquad i \in \mathcal{M} \quad t \in \mathcal{T}. \tag{26}$$

Constraints (22) establish that a customer can not be inserted into a route that already visits it, while constraints (23) impose that a customer cannot be removed from a route that does not visit it. Finally, constraints (24) 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 which differs from the corresponding one in s by at least two customers (i.e., at least two customers where removed or inserted, or one was removed and another one inserted), then the variation of the transportation cost in (21) 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 (6), (8) and (18) 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 $MIP_1(ML, s)$ and $MIP_2(ML, s)$.

2.2.2 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 [19] and [21]) 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 $v_i \in Z^+$ $(i = 1, \cdots, n)$, and given a capacity $C < \sum_{i=1}^{n} w_i$, determine a subset A of $\{1, \cdots, n\}$ with $\sum_{i \in A} w_i \leq C$ and of maximum value $\sum_{i \in A} v_i$.

Note that the knapsack problem remains NP-hard if it is assumed that $\frac{2}{3} \sum_{i=1}^{n} w_i \leq C < \sum_{i=1}^{n} w_i$. 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 $w_{n+1} = 2 \sum_{i=1}^{n} w_i - 3C + \delta$, $(0 \leq \delta)$, 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 since 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,

$$C' = \frac{2}{3}3(\sum_{i=1}^{n} w_i - C) + \delta = \frac{2}{3}(\sum_{i=1}^{n} w_i + (2\sum_{i=1}^{n} w_i - 3C + \delta)) + \frac{\delta}{3} = \frac{2}{3}\sum_{i=1}^{n+1} w_i + \frac{\delta}{3}.$$

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

Theorem 1 *The OU-Route-Assignment, OU-Customer-Assignment are both NP-hard.*

Proof Given an instance of the knapsack problem with n objects of weight w_i and size v_i $(i = 1, \cdots, n)$ and with capacity C such that $\frac{2}{3}\sum_{i=1}^{n} w_i \leq C < \sum_{i=1}^{n} w_i$, we construct an instance of the IRP where $\mathcal{M} = \{1, \cdots, n\}$, $\mathcal{T} = \{1, 2\}$, $r_{0t} = 0$ for $t \in \mathcal{T}$, $r_{it} = r_i = \frac{w_i}{3}$ for $i \in \mathcal{M}$ and $t \in \mathcal{T}$, $B_0 = \sum_{i \in M} 3r_i$, $h_0 = 0$, $I_{i0} = r_i$, $U_i = 3r_i$, $h_i = \frac{v_i}{r_i} > 0$ for all customers i, and with transportation capacity C. The above assumption about C translates to $\sum_{i \in \mathcal{M}} 2r_i \leq C < \sum_{i \in \mathcal{M}} 3r_i$. 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 since 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 $t = 1$ since 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 $4r_i$, while 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 stock-out 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 $\sum_{i \in \mathcal{M}} 2r_i \leq C$ in that case. Since 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. Since both routes in s contain all customers, no customer can be added to a route. Hence, the objective functions in (2) and (21) can be rewritten as follows, where $\mathcal{M}_t$ denotes the set of customers served at time t:

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

Since 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 value 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:

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

Since 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 [14]), which is defined as follows: given n objects of weight $w_i \in Z^+$, determine whether there exists a subset A of $\{1, \cdots, n\}$ with $\sum_{i\in A} w_i = \sum_{i\notin A} w_i$.

Theorem 2 *The ML-Route-Assignment and ML-Customer-Assignment are both NP-hard.*

Proof Given an instance of the partition problem with n objects of size w_i $(i = 1, \cdots, n)$, we consider an instance of the IRP with $\mathcal{M} = \{1, \cdots, n\}$, $\mathcal{T} = \{1, 2\}$, $r_{0t} = 0$ 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 with 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 $\frac{1}{2}w_i$ 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 since 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 $t = 1$ since 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 $w_i + x_{i1}$.

Since both routes in s contain all customers, no customer can be added to a route. Hence, since $h_0 = 0$, both (2) and (21) 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 which is visited in periods $t = 1$ and $t = 2$, then $\sum_{i\in\mathcal{M}}(w_i+x_{i1})-2Cm > \sum_{i\in\mathcal{M}} w_i - 2C(n-1) = 2C(2-n)$, while 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 $\sum_{i\in\mathcal{M}_1} w_i \leq C$ and $\sum_{i\in\mathcal{M}_2} w_i \leq C$. Since $C = \frac{1}{2}\sum_{i\in\mathcal{M}} w_i$, this is equivalent to say 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. □

2.2.3 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 in the solution and trying to reduce the transportation cost by applying the heuristic by Lin and Kernighan ([20]) for the Traveling Salesman Problem (TSP). We then apply to the new solution the MIPs described in Section 2.2.1. If the output of one of these MIPs differs from the input, we apply again the Lin and Kerninghan heuristic on each route that was modified. We first try to improve s_{best} by solving $MIP_1(\diamond, s_{best})$ (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 which are in r_2 but not in r_1 (using the cheapest insertion method), and by then optimizing the route using the Lin and Kerninghan 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 since a stock-out 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 $f(s_{best})$. We finally try to improve s_{best} by solving $MIP_2(\diamond, s_{best})$. A detailed description of the *Improvement* procedure can be found in Algorithm 3 in the Appendix.

2.3 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 $\frac{JumpIter}{2}$ iterations, i was never visited at time $t' \neq t$ during the last $\frac{JumpIter}{2}$ iterations, and the move of the visit to i from t to t'

does not create a stock-out 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 Section 2.2) and consider the resulting solution as the new current solution s for the tabu search.

3 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* model (MIP_1) and the *Route Merging* model (MIP_2) have been solved to optimality by using CPLEX 10.1.

The hybrid heuristic HAIR was tested on the benchmark instances used in [1] 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 in order 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: $\frac{3}{2} \sum_{i \in \mathcal{M}} r_i$;
- 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 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 used to determine the length of the tabu list in equation (1) (with $L = 10$). The value of the optimal solution 'z^*' is taken from [1]. Column HAIR reports the value of the solution given by HAIR while '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 [3]: column 'BPS' reports the solution value while '% error' indicates the error with respect to the optimal solution.

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 1-2). Both when $h_i \in [0.1, 0.5]$ and when $h_i \in [0.01, 0.05]$, HAIR finds all optimal solutions, except in one case (abs4n20.dat). In general, on 88% of the instances with short horizon $H = 3$ HAIR finds the optimal solution, while 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 $h_0 = 0.3$ have been chosen for these tests. The results are shown in Tables 9 and 10. 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}, "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 Section 2.2.3), "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.

We also tested HAIR on larger instances where the comparison with the exact solution cannot be made as 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 11 and 12. For each table, the first two columns report the name and the size of the instance. Then, the following 7 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 4 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 one hour, respectively. The following column gives the error of the solution generated by the heuristic presented in Bertazzi, Paletta and Speranza (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 3 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.

4 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 6 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.

Table 1: ML policy – Computational results on instances with $H = 3$ and $h_i \in [0.01, 0.05]$

instance	n	λ	z^*	HAIR	CPU	% error
abs1n5.dat	5	0.5	1235.92	1235.92	2	0.00
abs2n5.dat	5	0.5	988.66	988.66	2	0.00
abs3n5.dat	5	0.5	1758.02	1758.02	3	0.00
abs4n5.dat	5	0.5	1397.29	1397.29	4	0.00
abs5n5.dat	5	0.5	999.42	999.42	2	0.00
abs1n10.dat	10	0.5	1743.07	1743.07	10	0.00
abs2n10.dat	10	0.5	2229.25	2229.25	10	0.00
abs3n10.dat	10	0.5	1871.14	1871.14	9	0.00
abs4n10.dat	10	0.5	1773.00	1773.00	8	0.00
abs5n10.dat	10	0.5	1938.18	1938.18	10	0.00
abs1n15.dat	15	0.5	2131.04	2131.04	30	0.00
abs2n15.dat	15	0.5	2131.58	2131.58	24	0.00
abs3n15.dat	15	0.5	2463.68	2463.68	25	0.00
abs4n15.dat	15	0.5	2151.94	2151.94	34	0.00
abs5n15.dat	15	0.5	2160.59	2160.59	21	0.00
abs1n20.dat	20	0.5	2267.32	2267.32	43	0.00
abs2n20.dat	20	0.5	2497.90	2497.90	72	0.00
abs3n20.dat	20	0.5	2590.48	2590.48	60	0.00
abs4n20.dat	20	0.5	3122.31	3122.99	90	0.02
abs5n20.dat	20	0.5	2849.90	2849.90	58	0.00
abs1n25.dat	25	0.5	2840.92	2840.92	107	0.00
abs2n25.dat	25	0.5	3014.56	3014.56	92	0.00
abs3n25.dat	25	0.5	3050.40	3050.40	100	0.00
abs4n25.dat	25	0.5	3078.67	3078.67	161	0.00
abs5n25.dat	25	0.5	2954.96	2954.96	104	0.00
abs1n30.dat	30	0.5	3427.78	3427.78	221	0.00
abs2n30.dat	30	0.5	3328.94	3328.94	190	0.00
abs3n30.dat	30	0.5	3471.86	3471.86	249	0.00
abs4n30.dat	30	0.5	3321.48	3321.48	436	0.00
abs5n30.dat	30	0.5	2914.60	2914.60	201	0.00
abs1n35.dat	35	0.5	3346.12	3346.12	446	0.00
abs2n35.dat	35	0.5	3541.71	3541.71	601	0.00
abs3n35.dat	35	0.5	3811.78	3811.78	340	0.00
abs4n35.dat	35	0.5	3229.34	3229.34	402	0.00
abs5n35.dat	35	0.5	3315.26	3315.26	284	0.00
abs1n40.dat	40	0.5	3702.14	3702.14	589	0.00
abs2n40.dat	40	0.5	3832.09	3832.09	546	0.00
abs3n40.dat	40	0.5	3874.62	3874.62	471	0.00
abs4n40.dat	40	0.5	3534.80	3534.80	550	0.00
abs5n40.dat	40	0.5	3575.46	3575.46	450	0.00
abs1n45.dat	45	0.5	3950.86	3950.86	939	0.00
abs2n45.dat	45	0.5	3702.72	3702.72	700	0.00
abs3n45.dat	45	0.5	3968.04	3968.04	668	0.00
abs4n45.dat	45	0.5	3998.26	3998.26	837	0.00
abs5n45.dat	45	0.5	3717.54	3717.54	744	0.00
abs1n50.dat	50	0.5	4047.18	4047.18	1009	0.00
abs2n50.dat	50	0.5	4512.96	4512.96	1229	0.00
abs3n50.dat	50	0.5	4451.44	4451.44	1112	0.00
abs4n50.dat	50	0.5	4405.84	4405.84	1105	0.00
abs5n50.dat	50	0.5	4218.37	4218.37	982	0.00

Table 2: ML policy – Computational results on instances with $H = 3$ and $h_i \in [0.1, 0.5]$

instance	n	λ	z^*	HAIR	CPU	% error
abs1n5.dat	5	0.5	2108.34	2108.34	1	0.00
abs2n5.dat	5	0.5	1767.06	1767.06	1	0.00
abs3n5.dat	5	0.5	2973.00	2973.00	3	0.00
abs4n5.dat	5	0.5	1981.04	1981.04	4	0.00
abs5n5.dat	5	0.5	2170.04	2170.04	1	0.00
abs1n10.dat	10	0.5	4510.61	4510.61	10	0.00
abs2n10.dat	10	0.5	4504.61	4504.61	11	0.00
abs3n10.dat	10	0.5	4031.40	4031.40	9	0.00
abs4n10.dat	10	0.5	3933.46	3933.46	9	0.00
abs5n10.dat	10	0.5	4709.79	4709.79	10	0.00
abs1n15.dat	15	0.5	5589.70	5589.70	31	0.00
abs2n15.dat	15	0.5	5443.34	5443.34	38	0.00
abs3n15.dat	15	0.5	6300.86	6300.86	27	0.00
abs4n15.dat	15	0.5	4977.58	4977.58	44	0.00
abs5n15.dat	15	0.5	4867.53	4867.53	21	0.00
abs1n20.dat	20	0.5	6859.02	6859.02	44	0.00
abs2n20.dat	20	0.5	7087.74	7087.74	76	0.00
abs3n20.dat	20	0.5	7354.68	7354.68	68	0.00
abs4n20.dat	20	0.5	6952.79	6957.78	88	0.07
abs5n20.dat	20	0.5	7874.26	7874.26	57	0.00
abs1n25.dat	25	0.5	8227.86	8227.86	115	0.00
abs2n25.dat	25	0.5	8765.72	8765.72	102	0.00
abs3n25.dat	25	0.5	9382.42	9382.42	108	0.00
abs4n25.dat	25	0.5	8452.93	8452.93	165	0.00
abs5n25.dat	25	0.5	10081.42	10081.42	116	0.00
abs1n30.dat	30	0.5	12066.86	12066.86	234	0.00
abs2n30.dat	30	0.5	10941.32	10941.32	271	0.00
abs3n30.dat	30	0.5	12122.36	12122.36	245	0.00
abs4n30.dat	30	0.5	9687.10	9687.10	312	0.00
abs5n30.dat	30	0.5	9773.90	9773.90	209	0.00
abs1n35.dat	35	0.5	11659.88	11659.88	483	0.00
abs2n35.dat	35	0.5	10466.80	10466.80	448	0.00
abs3n35.dat	35	0.5	13776.46	13776.46	322	0.00
abs4n35.dat	35	0.5	10307.40	10307.40	447	0.00
abs5n35.dat	35	0.5	10847.82	10847.82	297	0.00
abs1n40.dat	40	0.5	13364.92	13364.92	624	0.00
abs2n40.dat	40	0.5	11317.85	11317.85	612	0.00
abs3n40.dat	40	0.5	13598.94	13598.94	544	0.00
abs4n40.dat	40	0.5	11353.39	11353.39	593	0.00
abs5n40.dat	40	0.5	13070.18	13070.18	523	0.00
abs1n45.dat	45	0.5	14179.10	14179.10	958	0.00
abs2n45.dat	45	0.5	13142.22	13142.22	746	0.00
abs3n45.dat	45	0.5	14843.60	14843.60	717	0.00
abs4n45.dat	45	0.5	13574.50	13574.50	876	0.00
abs5n45.dat	45	0.5	13587.26	13587.26	732	0.00
abs1n50.dat	50	0.5	14577.30	14577.30	1092	0.00
abs2n50.dat	50	0.5	15001.64	15001.64	1036	0.00
abs3n50.dat	50	0.5	15279.49	15279.49	1342	0.00
abs4n50.dat	50	0.5	16517.00	16517.00	1217	0.00
abs5n50.dat	50	0.5	15678.67	15678.67	1041	0.00

Table 3: ML policy – Computational results on instances with $H = 6$ and $h_i \in [0.01, 0.05]$

instance	n	λ	z^*	HAIR	CPU	% error
abs1n5.dat	5	0.5	3187.30	3187.3	17	0.00
abs2n5.dat	5	0.5	2565.92	2566.47	19	0.02
abs3n5.dat	5	0.5	4489.83	4489.83	19	0.00
abs4n5.dat	5	0.5	3174.35	3174.35	14	0.00
abs5n5.dat	5	0.5	2267.10	2267.1	11	0.00
abs1n10.dat	10	0.5	4141.53	4141.53	69	0.00
abs2n10.dat	10	0.5	5044.63	5046.13	69	0.03
abs3n10.dat	10	0.5	4506.83	4508.8	71	0.04
abs4n10.dat	10	0.5	4823.53	4823.53	49	0.00
abs5n10.dat	10	0.5	4545.98	4546.03	65	0.00
abs1n15.dat	15	0.5	5389.08	5391.17	166	0.04
abs2n15.dat	15	0.5	5418.47	5423.47	245	0.09
abs3n15.dat	15	0.5	5897.68	5897.68	161	0.00
abs4n15.dat	15	0.5	5335.01	5335.01	203	0.00
abs5n15.dat	15	0.5	5052.51	5074.68	127	0.44
abs1n20.dat	20	0.5	6114.04	6114.04	475	0.00
abs2n20.dat	20	0.5	5957.31	5958.29	561	0.02
abs3n20.dat	20	0.5	6784.06	6786.04	429	0.03
abs4n20.dat	20	0.5	7309.54	7394.84	508	1.17
abs5n20.dat	20	0.5	6961.82	6967.4	309	0.08
abs1n25.dat	25	0.5	7052.06	7064	1678	0.16
abs2n25.dat	25	0.5	7231.75	7254.38	725	0.31
abs3n25.dat	25	0.5	7514.57	7514.57	1162	0.00
abs4n25.dat	25	0.5	7462.08	7462.08	744	0.00
abs5n25.dat	25	0.5	7048.40	7059.9	845	0.16
abs1n30.dat	30	0.5	8052.73	8092.11	1930	0.49
abs2n30.dat	30	0.5	7629.99	7631.21	2602	0.02
abs3n30.dat	30	0.5	8136.21	8137.33	2422	0.01
abs4n30.dat	30	0.5	7502.49	7502.49	3712	0.00
abs5n30.dat	30	0.5	7228.63	7265.35	2003	0.51

Table 4: ML policy – Computational results on instances with $H = 6$ and $h_i \in [0.1, 0.5]$

instance	n	λ	z^*	HAIR	CPU	% error
abs1n5.dat	5	0.5	5789.35	5789.35	16	0.00
abs2n5.dat	5	0.5	4883.77	4883.77	14	0.00
abs3n5.dat	5	0.5	6643.29	6643.29	24	0.00
abs4n5.dat	5	0.5	5076.88	5076.88	22	0.00
abs5n5.dat	5	0.5	4377.71	4377.71	13	0.00
abs1n10.dat	10	0.5	8480.17	8480.17	86	0.00
abs2n10.dat	10	0.5	8347.44	8356.35	79	0.11
abs3n10.dat	10	0.5	8321.68	8360.05	107	0.46
abs4n10.dat	10	0.5	8474.26	8474.26	77	0.00
abs5n10.dat	10	0.5	9386.03	9386.03	82	0.00
abs1n15.dat	15	0.5	12052.56	12052.56	179	0.00
abs2n15.dat	15	0.5	11823.55	11825.87	243	0.02
abs3n15.dat	15	0.5	13305.71	13305.71	270	0.00
abs4n15.dat	15	0.5	10479.29	10494.11	239	0.14
abs5n15.dat	15	0.5	10054.09	10070.56	201	0.16
abs1n20.dat	20	0.5	14266.53	14301	450	0.24
abs2n20.dat	20	0.5	14477.84	14551.58	392	0.51
abs3n20.dat	20	0.5	14319.35	14328.68	426	0.07
abs4n20.dat	20	0.5	14390.27	14417	398	0.19
abs5n20.dat	20	0.5	15556.70	15563.44	341	0.04
abs1n25.dat	25	0.5	15487.66	15555.79	805	0.44
abs2n25.dat	25	0.5	16502.46	16516.45	795	0.08
abs3n25.dat	25	0.5	17833.35	17833.35	894	0.00
abs4n25.dat	25	0.5	16193.96	16316.37	1070	0.76
abs5n25.dat	25	0.5	18552.42	18570.92	1570	0.10
abs1n30.dat	30	0.5	22837.94	22874.4	1922	0.16
abs2n30.dat	30	0.5	19876.76	19997.28	1850	0.61
abs3n30.dat	30	0.5	23096.93	23198.29	2192	0.44
abs4n30.dat	30	0.5	17509.82	17550.33	1952	0.23
abs5n30.dat	30	0.5	18731.81	18770.28	1694	0.21

Table 5: OU policy – Computational results on instances 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	1281.68	1281.68	3	0.00	1281.68	0.00
abs2n5.dat	5	0.5	1176.63	1176.63	3	0.00	1176.63	0.00
abs3n5.dat	5	0.5	2020.65	2020.65	4	0.00	2178.94	7.83
abs4n5.dat	5	0.5	1449.43	1449.43	2	0.00	1449.43	0.00
abs5n5.dat	5	0.5	1165.40	1165.40	3	0.00	1242.07	6.58
abs1n10.dat	10	0.5	2167.37	2167.37	14	0.00	2198.56	1.44
abs2n10.dat	10	0.5	2510.13	2510.13	12	0.00	2510.13	0.00
abs3n10.dat	10	0.5	2099.68	2099.68	13	0.00	2099.68	0.00
abs4n10.dat	10	0.5	2188.01	2188.01	11	0.00	2188.01	0.00
abs5n10.dat	10	0.5	2178.15	2178.45	14	0.01	2231.67	2.46
abs1n15.dat	15	0.5	2236.53	2236.53	40	0.00	2271.68	1.57
abs2n15.dat	15	0.5	2506.21	2506.21	41	0.00	2506.21	0.00
abs3n15.dat	15	0.5	2841.06	2841.06	44	0.00	2841.06	0.00
abs4n15.dat	15	0.5	2430.07	2430.07	43	0.00	2632.18	8.32
abs5n15.dat	15	0.5	2453.50	2453.50	39	0.00	2524.95	2.91
abs1n20.dat	20	0.5	2793.29	2793.29	75	0.00	2793.29	0.00
abs2n20.dat	20	0.5	2799.90	2799.90	105	0.00	2873.51	2.63
abs3n20.dat	20	0.5	3101.60	3104.29	101	0.09	3163.94	2.01
abs4n20.dat	20	0.5	3239.31	3239.31	87	0.00	3239.31	0.00
abs5n20.dat	20	0.5	3330.99	3330.99	153	0.00	3814.54	14.52
abs1n25.dat	25	0.5	3309.64	3309.64	341	0.00	3624.49	9.51
abs2n25.dat	25	0.5	3495.97	3495.97	214	0.00	3544.55	1.39
abs3n25.dat	25	0.5	3481.45	3481.45	278	0.00	3622.70	4.06
abs4n25.dat	25	0.5	3272.74	3272.74	295	0.00	3272.74	0.00
abs5n25.dat	25	0.5	3695.94	3695.94	166	0.00	3695.94	0.00
abs1n30.dat	30	0.5	3918.76	3918.76	326	0.00	4022.30	2.64
abs2n30.dat	30	0.5	3737.11	3737.11	497	0.00	3874.22	3.67
abs3n30.dat	30	0.5	3761.85	3761.85	607	0.00	3931.85	4.52
abs4n30.dat	30	0.5	3532.47	3532.47	452	0.00	3676.90	4.09
abs5n30.dat	30	0.5	3265.89	3269.76	693	0.12	3365.76	3.06
abs1n35.dat	35	0.5	3694.48	3694.48	908	0.00	3737.48	1.16
abs2n35.dat	35	0.5	3796.80	3796.80	704	0.00	3960.39	4.31
abs3n35.dat	35	0.5	4351.09	4359.08	532	0.18	4665.40	7.22
abs4n35.dat	35	0.5	3766.39	3766.39	1225	0.00	3939.11	4.59
abs5n35.dat	35	0.5	3625.57	3625.57	675	0.00	3807.86	5.03
abs1n40.dat	40	0.5	4263.43	4274.71	903	0.26	4732.75	11.01
abs2n40.dat	40	0.5	4166.95	4166.95	1539	0.00	4415.23	5.96
abs3n40.dat	40	0.5	4337.30	4340.06	1310	0.06	4591.92	5.87
abs4n40.dat	40	0.5	3846.84	3846.84	1441	0.00	4000.81	4.00
abs5n40.dat	40	0.5	4013.98	4013.98	650	0.00	4234.01	5.48
abs1n45.dat	45	0.5	4369.38	4369.38	1493	0.00	4568.68	4.56
abs2n45.dat	45	0.5	4226.82	4226.82	1224	0.00	4655.80	10.15
abs3n45.dat	45	0.5	4317.08	4317.08	1904	0.00	4572.82	5.92
abs4n45.dat	45	0.5	4527.95	4559.36	1292	0.69	4962.14	9.59
abs5n45.dat	45	0.5	3911.82	3911.82	1387	0.00	4215.11	7.75
abs1n50.dat	50	1	4629.92	4670.41	1782	0.87	4884.83	5.51
abs2n50.dat	50	1	4919.75	4919.75	2004	0.00	5123.98	4.15
abs3n50.dat	50	1	4868.36	4868.36	2648	0.00	5269.43	8.24
abs4n50.dat	50	1	4972.25	5014.17	1843	0.84	5273.98	6.07
abs5n50.dat	50	1	4664.05	4687.16	3126	0.50	4901.19	5.08

Table 6: OU policy – Computational results on instances 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	2149.80	2149.80	4	0.00	2149.80	0.00
abs2n5.dat	5	0.5	1959.05	1959.05	4	0.00	1959.05	0.00
abs3n5.dat	5	0.5	3265.44	3265.44	7	0.00	3408.48	4.38
abs4n5.dat	5	0.5	2034.44	2034.44	2	0.00	2034.44	0.00
abs5n5.dat	5	0.5	2362.16	2362.16	6	0.00	2413.72	2.18
abs1n10.dat	10	0.5	4970.62	4970.62	15	0.00	5015.82	0.91
abs2n10.dat	10	0.5	4803.17	4803.17	13	0.00	5142.58	7.07
abs3n10.dat	10	0.5	4289.84	4289.84	13	0.00	4289.84	0.00
abs4n10.dat	10	0.5	4347.06	4347.06	11	0.00	4347.06	0.00
abs5n10.dat	10	0.5	5041.62	5044.44	14	0.06	5078.05	0.72
abs1n15.dat	15	0.5	5713.84	5713.84	37	0.00	5713.84	0.00
abs2n15.dat	15	0.5	5821.04	5822.88	41	0.03	5822.88	0.03
abs3n15.dat	15	0.5	6711.25	6711.25	45	0.00	6894.43	2.73
abs4n15.dat	15	0.5	5227.56	5227.56	49	0.00	5437.22	4.01
abs5n15.dat	15	0.5	5210.85	5215.02	61	0.08	5424.97	4.11
abs1n20.dat	20	0.5	7353.82	7353.82	108	0.00	7449.47	1.30
abs2n20.dat	20	0.5	7385.03	7385.03	101	0.00	7450.67	0.89
abs3n20.dat	20	0.5	7903.97	7907.40	96	0.04	7974.47	0.89
abs4n20.dat	20	0.5	7050.91	7050.91	143	0.00	7601.67	7.81
abs5n20.dat	20	0.5	8405.83	8405.83	73	0.00	8876.11	5.59
abs1n25.dat	25	0.5	8657.70	8657.70	205	0.00	8985.82	3.79
abs2n25.dat	25	0.5	9266.87	9266.87	164	0.00	9408.18	1.52
abs3n25.dat	25	0.5	9843.60	9843.60	208	0.00	9843.60	0.00
abs4n25.dat	25	0.5	8677.86	8677.86	342	0.00	8677.86	0.00
abs5n25.dat	25	0.5	10857.68	10857.68	191	0.00	10857.68	0.00
abs1n30.dat	30	0.5	12635.55	12635.55	345	0.00	12847.79	1.68
abs2n30.dat	30	0.5	11351.36	11351.36	273	0.00	11487.63	1.20
abs3n30.dat	30	0.5	12509.26	12613.46	621	0.83	12679.26	1.36
abs4n30.dat	30	0.5	9928.35	9928.35	573	0.00	10018.88	0.91
abs5n30.dat	30	0.5	10178.63	10181.69	346	0.03	10270.81	0.91
abs1n35.dat	35	0.5	11984.69	11984.69	600	0.00	12382.03	3.32
abs2n35.dat	35	0.5	10706.91	10706.91	1160	0.00	10998.26	2.72
abs3n35.dat	35	0.5	14411.58	14478.78	581	0.47	14732.69	2.23
abs4n35.dat	35	0.5	10844.98	10844.98	1202	0.00	11001.99	1.45
abs5n35.dat	35	0.5	11195.87	11195.87	624	0.00	11365.71	1.52
abs1n40.dat	40	0.5	14006.57	14006.57	931	0.00	14455.90	3.21
abs2n40.dat	40	0.5	11722.58	11722.58	996	0.00	11941.41	1.87
abs3n40.dat	40	0.5	14107.14	14107.14	2035	0.00	14655.81	3.89
abs4n40.dat	40	0.5	11684.43	11684.43	1276	0.00	11763.03	0.67
abs5n40.dat	40	0.5	13536.57	13536.57	1230	0.00	13759.25	1.65
abs1n45.dat	45	0.5	14661.20	14661.20	2099	0.00	14867.01	1.40
abs2n45.dat	45	0.5	13675.96	13675.96	1235	0.00	14161.64	3.55
abs3n45.dat	45	0.5	15316.57	15316.57	1416	0.00	15564.91	1.62
abs4n45.dat	45	0.5	14096.84	14121.05	1175	0.17	14537.57	3.13
abs5n45.dat	45	0.5	13838.54	13840.06	1747	0.01	14215.84	2.73
abs1n50.dat	50	1	15235.83	15235.83	3989	0.00	15543.49	2.02
abs2n50.dat	50	1	15453.78	15455.34	2412	0.01	15630.97	1.15
abs3n50.dat	50	1	15747.73	15867.45	2996	0.76	16055.08	1.95
abs4n50.dat	50	1	17163.52	17173.36	2169	0.06	17450.53	1.67
abs5n50.dat	50	1	16143.06	16143.06	2585	0.00	16313.73	1.06

Table 7: OU policy – Computational results on instances 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	3335.24	3335.24	16	0.00	3348.24	0.39
abs2n5.dat	5	0.5	2722.33	2722.33	23	0.00	2722.33	0.00
abs3n5.dat	5	0.5	4776.00	4776.00	17	0.00	4783.42	0.16
abs4n5.dat	5	0.5	3246.66	3246.66	21	0.00	3389.66	4.40
abs5n5.dat	5	0.5	2419.67	2419.67	12	0.00	2498.52	3.26
abs1n10.dat	10	0.5	4499.25	4499.25	78	0.00	4534.62	0.79
abs2n10.dat	10	0.5	5236.98	5236.98	105	0.00	5236.98	0.00
abs3n10.dat	10	0.5	4652.53	4652.53	67	0.00	4652.53	0.00
abs4n10.dat	10	0.5	5104.91	5104.91	73	0.00	5314.16	4.10
abs5n10.dat	10	0.5	4670.76	4670.76	61	0.00	4760.99	1.93
abs1n15.dat	15	0.5	5462.68	5462.68	415	0.00	5600.34	2.52
abs2n15.dat	15	0.5	5494.74	5494.74	360	0.00	5907.40	7.51
abs3n15.dat	15	0.5	6060.38	6060.38	276	0.00	6168.68	1.79
abs4n15.dat	15	0.5	5504.65	5504.65	380	0.00	6027.52	9.50
abs5n15.dat	15	0.5	5309.48	5309.48	256	0.00	5311.46	0.04
abs1n20.dat	20	2	6490.18	6501.26	559	0.17	6541.11	0.78
abs2n20.dat	20	2	6082.54	6093.54	987	0.18	6348.07	4.37
abs3n20.dat	20	2	6950.20	6953.78	733	0.05	7186.46	3.40
abs4n20.dat	20	2	7432.78	7432.78	698	0.00	7729.58	3.99
abs5n20.dat	20	2	7210.73	7210.73	1212	0.00	7369.90	2.21
abs1n25.dat	25	2	7095.86	7095.86	1495	0.00	7595.85	7.05
abs2n25.dat	25	2	7484.84	7504.84	1188	0.27	8067.75	7.79
abs3n25.dat	25	2	7728.76	7728.76	3170	0.00	7996.62	3.47
abs4n25.dat	25	2	7509.02	7509.02	1251	0.00	8035.38	7.01
abs5n25.dat	25	2	7452.28	7518.61	1496	0.89	7871.75	5.63
abs1n30.dat	30	2	8319.59	8349.59	2334	0.36	8761.13	5.31
abs2n30.dat	30	2	7761.53	7768.53	3251	0.09	8049.36	3.71
abs3n30.dat	30	2	8214.55	8365.82	3654	1.84	8654.55	5.36
abs4n30.dat	30	2	7574.80	7574.80	5146	0.00	8067.12	6.50
abs5n30.dat	30	2	7366.47	7402.71	2220	0.49	7538.91	2.34

Table 8: OU policy – Computational results on instances 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	5942.82	5942.82	17	0.00	5942.82	0.00
abs2n5.dat	5	0.5	5045.91	5045.91	20	0.00	5047.79	0.04
abs3n5.dat	5	0.5	6956.28	6956.28	16	0.00	6969.76	0.19
abs4n5.dat	5	0.5	5163.42	5163.42	29	0.00	5226.16	1.22
abs5n5.dat	5	0.5	4581.66	4581.66	17	0.00	4593.05	0.25
abs1n10.dat	10	0.5	8870.15	8870.15	78	0.00	8945.11	0.85
abs2n10.dat	10	0.5	8569.73	8569.73	66	0.00	8999.23	5.01
abs3n10.dat	10	0.5	8509.81	8509.81	129	0.00	8509.81	0.00
abs4n10.dat	10	0.5	8792.29	8792.29	112	0.00	8994.79	2.30
abs5n10.dat	10	0.5	9620.07	9620.07	67	0.00	9735.40	1.20
abs1n15.dat	15	0.5	12118.83	12118.83	266	0.00	12118.83	0.00
abs2n15.dat	15	0.5	11932.10	11932.1	305	0.00	12158.12	1.89
abs3n15.dat	15	0.5	13554.15	13554.15	369	0.00	13554.15	0.00
abs4n15.dat	15	0.5	10618.55	10618.55	285	0.00	10711.77	0.88
abs5n15.dat	15	0.5	10385.54	10385.54	220	0.00	10721.92	3.24
abs1n20.dat	20	2	14702.95	14722.95	507	0.14	15250.81	3.73
abs2n20.dat	20	2	14646.96	14670.34	746	0.16	14785.04	0.94
abs3n20.dat	20	2	14532.91	14577.14	863	0.30	14764.08	1.59
abs4n20.dat	20	2	14539.72	14539.72	1084	0.00	14590.96	0.35
abs5n20.dat	20	2	15896.71	15904	533	0.04	16506.45	3.84
abs1n25.dat	25	2	15581.47	15610.98	1941	0.19	15726.50	0.93
abs2n25.dat	25	2	16823.16	16843.16	1544	0.12	17017.02	1.15
abs3n25.dat	25	2	18098.02	18121.26	2198	0.13	18506.33	2.26
abs4n25.dat	25	2	16303.69	16303.69	1368	0.00	17026.13	4.43
abs5n25.dat	25	2	19047.70	19080.27	1856	0.17	19394.10	1.82
abs1n30.dat	30	2	23183.99	23183.99	2365	0.00	23754.83	2.46
abs2n30.dat	30	2	20090.29	20159.96	2178	0.35	20713.82	3.10
abs3n30.dat	30	2	23382.73	23439.91	3978	0.24	23918.93	2.29
abs4n30.dat	30	2	17649.53	17746.79	5063	0.55	18247.06	3.39
abs5n30.dat	30	2	18979.93	18997.61	2239	0.09	19270.91	1.53

Table 9: ML policy – Impact of the improvement phase

	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	1922	0.16	2506	1.73	1500	2.51	1870	0.17	1326	2.50	2111	0.17	1338	2.50	1862	0.16
abs2n30.dat	1850	0.61	2125	0.75	2114	0.29	2107	1.00	1848	0.32	2968	0.93	2698	0.06	3654	0.43
abs3n30.dat	2192	0.44	1162	1.47	1193	1.49	1172	1.49	1196	1.46	1186	1.49	1184	1.47	1223	1.46
abs4n30.dat	1952	0.23	3571	0.79	5292	1.19	1991	0.30	3582	0.78	2030	0.30	3632	0.78	1966	0.23
abs5n30.dat	1694	0.21	4428	0.89	2448	0.44	1787	0.21	2663	0.28	1809	0.21	3259	0.05	1806	0.21

Table 10: OU policy – Impact of the improvement phase

	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	2365	0.00	8785	2.45	3648	3.09	1831	0.96	4188	2.36	2913	0.74	4992	2.36	2421	0.74
abs2n30.dat	2178	0.35	6612	0.00	2872	0.00	1995	0.03	2790	0.00	4562	0.00	2250	0.00	2361	0.00
abs3n30.dat	3978	0.24	2701	1.07	6365	0.09	3460	0.09	3392	0.99	3476	0.61	3550	0.98	3071	0.67
abs4n30.dat	5063	0.55	6352	0.21	3105	1.71	3129	0.43	4608	1.74	3191	0.43	4340	1.61	2760	0.91
abs5n30.dat	2239	0.09	4556	1.12	4643	0.03	2263	0.03	4663	1.12	2981	0.13	2374	1.29	3095	0.03

Table 11: Computational results on instances with $n = \{50, 100, 200\}$ and $h_i \in [0.1, 0.5]$

		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	31147.82	0.00	0.00	0.00	0.00	1.55	0.5	30225.36	0.00	0.00	0.00
abs2n50.dat	50	4	30192.51	0.17	0.17	0.00	0.00	1.85	0.5	29856.26	0.00	0.00	0.00
abs3n50.dat	50	4	30420.96	0.00	0.00	0.00	0.00	2.17	0.5	29904.15	0.00	0.00	0.00
abs4n50.dat	50	4	31898.84	0.91	0.00	0.00	0.00	2.19	0.5	31677.87	0.18	0.00	0.00
abs5n50.dat	50	4	29518.68	0.83	0.00	0.00	0.00	0.56	0.5	29400.33	0.00	0.00	0.00
abs6n50.dat	50	4	32394.50	1.50	0.00	0.00	0.00	1.42	0.5	31946.33	0.78	0.78	0.78
abs7n50.dat	50	4	30165.00	1.52	1.52	0.29	0.00	3.06	0.5	29768.03	0.00	0.00	0.00
abs8n50.dat	50	4	26416.46	1.20	1.20	0.05	0.05	0.00	0.5	26521.96	0.00	0.00	0.00
abs9n50.dat	50	4	30671.88	1.52	0.00	0.00	0.00	1.94	0.5	30283.9	0.00	0.00	0.00
abs10n50.dat	50	4	32362.01	0.00	0.00	0.00	0.00	1.71	0.5	31397.84	0.00	0.00	0.00
abs1n100.dat	100	4	57962.58	0.34	0.33	0.00	0.00	2.58	0.5	57721.23	0.03	0.00	0.00
abs2n100.dat	100	4	53942.07	1.58	1.58	0.82	0.00	2.72	0.5	53432.8	0.00	0.00	0.00
abs3n100.dat	100	4	59370.69	1.72	1.37	1.36	0.00	0.70	0.5	58598.93	0.08	0.00	0.00
abs4n100.dat	100	4	52419.33	1.58	1.37	0.00	0.00	1.68	0.5	52030.59	0.00	0.00	0.00
abs5n100.dat	100	4	58679.72	1.54	1.40	0.47	0.00	0.59	0.5	58258.92	0.32	0.00	0.00
abs6n100.dat	100	4	55756.46	1.58	1.56	1.56	0.00	1.88	0.5	55280.01	0.37	0.34	0.00
abs7n100.dat	100	4	56861.23	0.16	0.03	0.00	0.00	1.64	0.5	56398.19	0.00	0.00	0.00
abs8n100.dat	100	4	55741.08	1.06	1.05	0.42	0.00	2.33	0.5	55384.47	0.61	0.61	0.61
abs9n100.dat	100	4	59219.73	1.63	1.52	1.08	0.00	2.45	0.5	58729.94	0.00	0.00	0.00
abs10n100.dat	100	4	57094.63	0.51	0.51	0.00	0.00	2.10	0.5	56644.22	0.38	0.00	0.00
abs1n200.dat	200	4	112090.80	0.46	0.46	0.08	0.00	0.47	0.5	110790.39	0.02	0.01	0.00
abs2n200.dat	200	4	114647.69	0.03	0.03	0.00	0.00	0.55	0.5	112401.98	0.34	0.34	0.04
abs3n200.dat	200	4	109339.50	0.68	0.68	0.68	0.00	1.01	0.5	108119.61	0.00	0.00	0.00
abs4n200.dat	200	4	110841.13	0.10	0.10	0.02	0.00	0.36	0.5	109309.48	0.06	0.06	0.00
abs5n200.dat	200	4	110994.17	0.00	0.00	0.00	0.00	0.38	0.5	109083.07	0.11	0.11	0.02
abs6n200.dat	200	4	110433.85	0.26	0.26	0.00	0.00	0.93	0.5	109071.2	0.16	0.01	0.00
abs7n200.dat	200	4	98880.57	0.71	0.71	0.15	0.00	1.06	0.5	97749.52	0.08	0.08	0.00
abs8n200.dat	200	4	104334.63	0.08	0.08	0.05	0.00	0.10	0.5	102194.63	0.10	0.02	0.00
abs9n200.dat	200	4	106994.52	0.18	0.18	0.08	0.00	0.14	0.5	104877.49	0.22	0.22	0.00
abs10n200.dat	200	4	110540.23	0.44	0.44	0.32	0.00	0.63	0.5	109045.17	0.19	0.08	0.02

Table 12: Computational results on instances with $n = \{50, 100, 200\}$ and $h_i \in [0.01, 0.05]$

		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	10409.13	4.44	4.44	0.00	0.00	5.74	0.5	9974.38	0.00	0.00	0.00
abs2n50.dat	50	4	10881.35	5.32	0.00	0.00	0.00	5.31	0.5	10632.24	0.58	0.00	0.00
abs3n50.dat	50	4	10767.39	0.00	0.00	0.00	0.00	6.63	0.5	10548.98	0.00	0.00	0.00
abs4n50.dat	50	4	10656.21	4.88	4.88	0.00	0.00	10.30	0.5	10555.38	0.00	0.00	0.00
abs5n50.dat	50	4	10234.60	1.57	1.57	0.00	0.00	8.57	0.5	10137.4	0.00	0.00	0.00
abs6n50.dat	50	4	10533.63	0.86	0.86	0.86	0.00	5.79	0.5	10166.1	0.00	0.00	0.00
abs7n50.dat	50	4	10460.82	3.67	3.67	3.29	2.37	0.00	0.5	10012.68	0.01	0.00	0.00
abs8n50.dat	50	4	10411.20	4.83	2.49	2.49	0.00	9.48	0.5	10547.97	2.35	2.35	0.00
abs9n50.dat	50	4	10305.69	1.63	1.63	1.63	0.00	2.49	0.5	10052.78	0.87	0.87	0.00
abs10n50.dat	50	4	10470.63	0.00	0.00	0.00	0.00	1.00	0.5	9727.74	0.00	0.00	0.00
abs1n100.dat	100	4	15938.80	0.44	0.03	0.03	0.00	4.38	0.5	15784.02	0.07	0.00	0.00
abs2n100.dat	100	4	14920.94	5.61	5.61	0.00	0.00	3.01	0.5	14754.1	0.00	0.00	0.00
abs3n100.dat	100	4	15836.71	7.44	6.07	5.82	0.00	8.35	0.5	15649.44	0.38	0.00	0.00
abs4n100.dat	100	4	15097.32	1.42	0.67	0.00	0.00	4.39	0.5	14755.76	0.34	0.00	0.00
abs5n100.dat	100	4	15697.01	4.84	4.84	0.00	0.00	1.49	0.5	15412.83	0.48	0.00	0.00
abs6n100.dat	100	4	15549.57	5.62	5.46	5.09	0.00	3.38	0.5	15327.08	0.95	0.58	0.00
abs7n100.dat	100	4	15725.24	0.54	0.00	0.00	0.00	3.59	0.5	15468.98	0.66	0.58	0.00
abs8n100.dat	100	4	15280.84	4.29	4.29	4.29	0.00	6.26	0.5	15078.16	1.95	0.22	0.00
abs9n100.dat	100	4	16471.32	0.06	0.01	0.00	0.00	2.69	0.5	15628.66	0.00	0.00	0.00
abs10n100.dat	100	4	15618.39	4.27	3.55	1.72	0.00	4.90	0.5	15495.87	0.07	0.00	0.00
abs1n200.dat	200	4	25690.21	0.01	0.01	0.01	0.00	1.06	0.5	24353.4	0.57	0.57	0.00
abs2n200.dat	200	4	25252.21	3.89	3.89	3.89	0.00	4.93	0.5	24576.55	0.45	0.45	0.14
abs3n200.dat	200	4	25170.57	1.07	1.07	0.43	0.43	0.00	0.5	23861.65	0.81	0.81	0.00
abs4n200.dat	200	4	24607.65	3.82	3.82	3.82	0.00	4.76	0.5	24232.73	1.41	1.41	0.40
abs5n200.dat	200	4	25263.32	1.57	1.57	1.55	1.40	0.00	0.5	24193.98	0.72	0.72	0.00
abs6n200.dat	200	4	24213.10	0.17	0.17	0.04	0.00	5.72	0.5	23651.81	0.95	0.95	0.00
abs7n200.dat	200	4	24731.63	0.02	0.02	0.00	0.00	1.02	0.5	23527.12	0.47	0.47	0.00
abs8n200.dat	200	4	23617.79	1.29	1.29	0.35	0.00	5.79	0.5	23230.42	0.81	0.81	0.35
abs9n200.dat	200	4	25387.67	0.39	0.39	0.35	0.00	0.14	0.5	23846.19	1.20	1.20	0.00
abs10n200.dat	200	4	23937.25	2.57	1.12	0.61	0.00	2.54	0.5	23609.34	0.61	0.61	0.00

References

[1] Archetti, C., Bertazzi, L., Laporte, G., Speranza, M.G. (2007), A branch-and-cut algorithm for a Vendor Managed Inventory routing problem, *Transportation Science* 41, 382-391.

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

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

[4] Bertazzi, L., Speranza, M.G., Savelsbergh, M.W.P. (2008), Inventory routing, in: *Vehicle routing: Latest advances and new challenges*, Golden, B., Raghavan, R., Wasil, E. (eds.), Springer, 49-72.

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

[6] M. Boudia, C. Prins, A memetic algorithm with dynamic population management for an integrated production-distribution problem, *European Journal of Operational Research*, doi :10.1016/j.ejor.2007.07.034.

[7] M. Boudia, M.A.O. Louly, C. Prins (2008), Fast heuristics for a combined production planning and vehicle routing problem, *Production Planning and Control* 19, 85-96.

[8] M. Boudia, M.A.O. Louly, C. Prins (2007), A reactive GRASP and path relinking for a combined production-distribution problem, *Computers and Operations Research* 34, 3402-3419.

[9] Campbell, A.M., Clarke, L., Kleywegt, A., Savelsbergh, M.W.P. (1998), The inventory routing problem, in: *Fleet Management and Logistics*, Crainic, T.G., Laporte, G. (eds.), 95-113, Kluwer, Boston.

[10] Cordeau, J.-F., Laporte, G., Savelsbergh, M.W.P., Vigo, D. (2009), Short-haul routing, in: *Handbooks in Operations Research and Management Science: Transportation*, Laporte, G., Barnhart, C. (eds.), to appear.

[11] Dror, M., Ball, M., Golden, B. (1985), A Computational comparison of algorithms for the inventory routing problem, *Annals of Operations Research* 4, 3-23.

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

[13] Federgruen, A., Zipkin, P. (1984), A combined vehicle routing and inventory allocation problem, *Operations Research* 32, 1019-1032.

[14] Garey, M.R. and Johnson, D.S. (1979), *Computers and intractability: A guide to the theory of NP-completeness*, W.H. Freeman and Company, New York.

[15] Gendreau, M., Hertz, A., Laporte, G. (1994), A tabu search heuristic for the vehicle routing problem, *Management Science* 40, 1276-1290.

[16] Glover, F. (1986), Future paths for integer programming and links to artificial intelligence, *Computers and Operations Research* 13, 533-549.

[17] Glover, F., Laguna, M. (1997), *Tabu search*, Kluwer Academic Publishers, Dordrecht.

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

[19] Kellerer, H., Pferschy, U. and Pisinger, D. (2004), *Knapsack Problems*, Springer–Verlag, Berlin-Heidelberg.

[20] Lin, S., Kernighan, B. W. (1973), An effective heuristic algorithm for the Traveling Salesman Problem. *Operations Research* 21, 498-516.

[21] Martello, S., Toth., P. (1990), *Knapsack problems: Algorithms and computer implementations*, Wiley, New York.

Appendix

In this appendix, we formally describe the construction of the neighborhood $N(s)$ and the *Improvement* procedure.

Algorithm 2 Construction of the neighborhood $N(s)$

Build $N'(s)$ by using the four simple types of changes on s and set $N(s) \leftarrow \emptyset$;
for all solutions $s' \in N'(s)$ **do**
  Determine the set $\mathcal{A}$ of customers i with $T_i(s) \neq T_i(s')$.
  while $\mathcal{A}$ is not empty **do**
    Choose a customer $i \in \mathcal{A}$ and remove it from $\mathcal{A}$.
    for all visit times $t \in T_i(s')$ **do**
      for all customers j served at time t in s' and such that $t \in T_j(s')$ **do**
        if $h_j > h_0$, $Q_t(s') > C$ or $B_t(s') < 0$ **then**

          OU policy:
          Let s'' be the solution obtained from s' by removing the visit to j at time t.
          if s'' is admissible and $f(s'') < f(s')$ **then**
            Set $s' \leftarrow s''$ and add j to $\mathcal{A}$.
          end if

          ML policy:
          Let $y \leftarrow \min\{x_{jt}, \min_{t'>t} I_{jt'}\}$.
          Let s'' be the solution obtained from s' by removing y units of delivery to j at time t (the visit to j at time t is removed if $y = x_{jt}$).
          if $f(s'') < f(s')$ **then**
            Set $s' \leftarrow s''$ and add j to $\mathcal{A}$ if j is not visited at time t in s'.
          end if

        end if
      end for

      ML policy:
      for all customers j served at time t in s' **do**
        if $h_j < h_0$ **then**
          Let $y \leftarrow \max_{t' \geq t}(I_{jt'} + x_{jt'})$.
          Let s'' be the solution obtained from s' by adding $U_j - y$ units of delivery to j at time t.
          if $f(s'') < f(s')$ **then**
            Set $s' \leftarrow s''$.
          end if
        end if
      end for

    end for
  end while
  Add s' to $N(s)$.
end for

Algorithm 3 Improvement Procedure

Set continue $\leftarrow$ true.
Set $s_{best} \leftarrow \mathbf{LK}(s_{best})$
while continue **do**
  Set continue $\leftarrow$ false.
  In the case of the OU policy, set $\diamond \leftarrow$OU, else set $\diamond \leftarrow$ML.

  (First type of improvement *)*
  Let s' be an optimal solution of $MIP_1(\diamond, s_{best})$. Set $s' \leftarrow \mathbf{LK}(s_{best}, s')$.
  if $f(s') < f(s_{best})$ **then** Set $s_{best} \leftarrow s'$ and continue $\leftarrow$ true.

  (Second type of improvement *)*
  Set $s_{merge} \leftarrow s_{best}$.
  Determine the set L of all pairs (r_1, r_2) of consecutive routes in s_{best}.
  for all pairs $(r_1, r_2) \in L$ **do**
    Let s_1 be the solution obtained from s_{best} by merging r_1 and r_2 into a single route r assigned to the same time as r_1.
    if $MIP_2(\diamond, s_1)$ is infeasible and r is not the last route in s_1 **then** Modify s_1 by anticipating the first route after r by one time period.
    if $MIP_2(\diamond, s_1)$ is feasible **then**
      Let s' be an optimal solution of $MIP_2(\diamond, s_1)$. Set $s' \leftarrow \mathbf{LK}(s_1, s')$.
      if $f(s') < f(s_{merge})$ **then** Set $s_{merge} \leftarrow s'$.
    end if
    Let s_2 be the solution obtained from s_{best} by merging r_1 and r_2 into a single route r assigned to the same time as r_2.
    if $MIP_2(\diamond, s_2)$ is infeasible and r is not the first route in s_2 **then** Modify s_2 by delaying the last route before r by one time period.
    if $MIP_2(\diamond, s_2)$ is feasible **then**
      Let s' be an optimal solution of $MIP_2(\diamond, s_2)$. Set $s' \leftarrow \mathbf{LK}(s_2, s')$.
      if $f(s') < f(s_{merge})$ **then** Set $s_{merge} \leftarrow s'$.
    end if
  end for
  if $f(s_{merge}) < f(s_{best})$ **then** Set $s_{best} \leftarrow s'$ and continue $\leftarrow$ true.

  (Third type of improvement *)*
  Let s' be an optimal solution of $MIP_2(\diamond, s_{best})$. Set $s' \leftarrow \mathbf{LK}(s_{best}, s')$.
  if $f(s') < f(s_{best})$ **then** Set $s_{best} \leftarrow s'$ and continue $\leftarrow$ true.
end while

Procedure LK(Input: s, s')
if $s = s'$ **then** Return s' **else** Return the solution obtained from s' by applying the Lin and Kerninghan heuristic on each route in s' that is different from the routes in s.