A Two-Stage Matheuristic Algorithm for Classical Inventory Routing Problem

Zhouxing Su, Shihao Huang, Chungen Li and Zhipeng Lü*

SMART, School of Computer Science and Technology, Huazhong University of Science and Technology, China
zhipeng.lv@hust.edu.cn

Abstract

The inventory routing problem (IRP), which is NP-hard, tackles the combination of inventory management and transportation optimization in supply chains. It seeks a minimum-cost schedule which utilizes a single vehicle to perform deliveries in multiple periods, so that no customer runs out of stock. Specifically, the solution of IRP can be represented as how many products should be delivered to which customer during each period, as well as the route in each period. We propose a two-stage matheuristic (TSMH) algorithm to solve the IRP. The first stage optimizes the overall schedule and generates an initial solution by a relax-and-repair method. The second stage employs an iterated tabu search procedure to achieve a fine-grained optimization to the current solution. Tested on 220 most commonly used benchmark instances, TSMH obtains advantages comparing to the state-of-the-art algorithms. The experimental results show that the proposed algorithm can obtain not only the optimal solutions for most small instances, but also better upper bounds for 40 out of 60 large instances. These results demonstrate that the TSMH algorithm is effective and efficient in solving the IRP. In addition, the comparative experiments justify the importance of two optimization stages of TSMH.

1 Introduction

Traditionally, suppliers or retailers manage their own warehouses and the inventory replenishment by themselves. They work out specific orders according to their production and consumption, and submit them to the logistics companies and the latter ones just do the deliveries. Due to the uncertainty of the call-in orders, it is hard for the logistics companies to make reasonable plans, which usually affects the operation efficiency and introduces unnecessary costs. In order to tackle the drawbacks brought by asymmetric information and achieve global optimization, vendor managed inventory (VMI) comes into play. Distinguished from the call-in mode, vendors (logistics companies) provide integrated management for all their customers (suppliers and retailers) in VMI mode. In other words, vendors are in charge of monitoring their clients’ inventory levels, predicting product consumption, scheduling replenishment, and planning delivery routes. It is believed that the integration of information and resources will bring benefits for both vendors and customers.

Inventory routing problem (IRP) is one of the most important challenges to apply the VMI mode in supply chain management. It combines two classical optimization problems, which are inventory problem and vehicle routing problem, respectively. Regarding the inventory part, it aims to minimize product backlog. Due to the risks for storing too many products that the commodities may expire, evaporate, be stolen, or encounter other accidents, there will be inventory holding costs proportional to the inventory level at each customer. As for the routing part, it requires to reduce the traveling costs which are usually measured by the total distance of the delivery routes. When the inventory management and the transportation optimization is jointly considered in the IRP, the entire planning horizon is divided into multiple independent periods, we need to decide the amount of product to replenish for each customer in each period, along with the delivery route in each period, so that no customer runs out of stock and the total cost for backlog and transportation is minimized. These two factors often contradict to each other, because less backlog means less quantity per delivery, which means more frequent deliveries if the consumption holds, which increases the traveling costs, and vice versa. Therefore, it is natural to consider the inventory management and the vehicle routing together, in order to balance and optimize the total costs.

As a challenging NP-hard problem [Coelho et al., 2013] with valuable applications, IRP has been drawing attentions
from both industrial and academic communities. Commercial software such as Aspen Fleet Optimizer was developed and widely used in the petroleum industry [Becraft et al., 2012]. Air Liquide proposed a complex inventory routing problem in its healthcare business as the topic of the ROADEF/EURO Challenge 2016\(^1\). It is also one of the topics of the 12th DIMACS Implementation Challenge\(^2\). Apart from the industrial applications, many researchers investigated the IRP and its variants in recent decades. There are mainly four characteristics that distinguishes different versions of IRPs, which are fleet composition, planning horizon, demand mode, and replenishment policy [Coelho et al., 2013], respectively. The single-vehicle multi-period deterministic-demand inventory routing problem under maximum-level policy proposed in Bertazzi et al. [2002] is one of the most representative and classical versions [Archetti et al., 2007; Archetti et al., 2012; Coelho et al., 2012; Alvarez et al., 2018; Chitsaz et al., 2019]. Based on this well-known IRP, several variants concerning homogeneous fleet [Coelho and Laporte, 2013; Adulyasak et al., 2013; Coelho and Laporte, 2014; Archetti et al., 2017] or multiple depots [Bertazzi et al., 2019] have been widely studied. Regarding the solution methods, several exact algorithms, heuristics, and hybrid algorithms were proposed for solving the IRP. The exact algorithms such as branch-and-cut algorithms [Archetti et al., 2007; Coelho and Laporte, 2013; Avella et al., 2017] have made great progress on solving small IRP instances. Desaulniers et al. [2015] presented a branch-cut-and-price algorithm for the IRP, which utilizes the column generation approach to calculate the lower bound, and employs the branch-and-cut algorithm to tighten the bound. In addition, several metaheuristic algorithms were proposed to tackle the IRP. Aksen et al. [2014] defined multiple neighborhood structures and integrated them into an adaptive large neighborhood search algorithm to solve the IRP. Park et al. [2016] presented a genetic algorithm inspired by the CPLEX optimizer. Iterated local search and simulated annealing were also used for tackling the IRP [Alvarez et al., 2018], and they can obtain satisfactory solutions in relatively short time.

Recently, more and more hybrid algorithms demonstrate their effectiveness in solving this intractable problem. These algorithms usually decompose complex problems into several sub-problems, and utilizes different kinds of methods for solving these reduced problems in certain hierarchy. The matheuristics, which integrate metaheuristic algorithms and mathematical programming techniques, is one of the most common and successful hybrid algorithms. For instance, Archetti et al. [2012] proposed a hybrid heuristic for the IRP, which uses the tabu search to find good solutions, and refines these solutions by optimizing two mixed-integer programming (MIP) models. Coelho et al. [2012] presented an algorithm based on adaptive large neighborhood search framework. It adopts a linear programming (LP) model to determine delivery quantities for each neighboring solution, and tunes the best solution found by another MIP model in a specified frequency. Campbell and Savelsbergh [2004] decomposed the IRP into two phases. It first constructs high-level base plan using an integer programming model, then utilizes an insertion heuristic to generate the complete schedule. In addition, Cordeau et al. [2015] solved the IRP by a multi-phase hybrid algorithm. It determines replenishment plans by a Lagrangian-based method in the first phase, generates vehicle routes by a constructive heuristic in the second phase, and reoptimizes the solution via MIP.

This paper proposes a two-stage matheuristic (TSMH) algorithm which consists of two optimization stages in different granularities for the classical IRP. The TSMH algorithm begins with a coarse-grained structural optimization which determines the initial replenishment schedule. In the second stage, the TSMH algorithm adopts a solution-based tabu search procedure to bring sophisticated refinement to the delivery routes, timing, and quantities. In both stages, the proposed algorithm employs both metaheuristics and mathematical programming to tackle the routing and inventory sub-problems, respectively. Tested on two sets of totally 220 widely used instances of the IRP, the proposed TSMH algorithm improves the best known results on 40 out of 60 large-scale instances and matches the records on most small instances. Moreover, comprehensive tests and comparisons demonstrate that the combination of the two stages and corresponding techniques are essential for the proposed algorithm.

\section{Classical Inventory Routing Problem}

The classical inventory routing problem is to deliver products to a number of clients in multiple periods. It aims to minimize the total inventory holding costs and vehicle traveling costs, while preventing any customer from running out of stock in each period. In order to accomplish this goal, a solution method needs to determine when, where, and how many products at each delivery, as well as the sequence of visits to each client in each period, as illustrated in Figure 1.

More precisely, the IRP problem is defined on an undirected complete graph \(G = \{V, E\}\). \(V = \{0, 1, ..., n\}\) is the set of vertices, where vertex 0 stands for the depot and \(V' = V - \{0\}\) represents the client set. \(E = \{(i, j) : i, j \in V, i \neq j\}\) denotes the edge set, where a corresponding traveling cost \(c_{ij}\) is associated with each edge \((i, j)\). There are totally \(p\) periods in the planning horizon, and the capacity of

\footnotesize
\begin{itemize}
  \item \(^1\)http://www.roadef.org/challenge/2016/en/sujet.php
  \item \(^2\)http://dimacs.rutgers.edu/events/details?eID=1090
\end{itemize}
the vehicle is $Q$. The inventory holding cost and maximum inventory level for each depot or client $i \in V$ are $H_i$ and $U_i$, respectively. Within each period $t \in T = \{1, ..., p\}$, the depot produces $r_{it}$ units of products and client $i \in V'$ consumes $r_{it}'$ units of products. To keep it simple and clear, we assume that the production and delivery always happen before loading and consumption at each vertex, respectively.

This simplification follows the same convention in the previous works [Archetti et al., 2007; Archetti et al., 2012; Coelho et al., 2012]. We define variable $x_{it}$ to be the quantity of products delivered to vertex $i \in V$ in period $t \in T$. Obviously, $x_{it}$ is always non-positive since the vehicle can only load instead of delivering products at depot. We use Boolean variable $y_{ij}$ to denote if edge $(i, j)$ exists in the delivery route in period $t \in T$. Let variable $I_i^t$ be the inventory level of vertex $i \in V$ at the end of period $t \in T \cup \{0\}$, except that the initial inventory level $I_i^0$ is already known. Then, the classical IRP can be formulated as the following MIP model.

$$\min \sum_{t \in T \cup \{0\}} \sum_{i \in V} H_i I_i^t + \sum_{t \in T} \sum_{i \in V} \sum_{j \in V, j \neq i} C_{ij} y_{ij},$$

subject to

$$\sum_{j \in V, j \neq i} y_{ij} - \sum_{j \in V, j \neq i} y_{ij} \leq 1, \forall i \in V, \forall t \in T,$$

$$\sum_{i, j \in S} y_{ij} \leq |S| - 1, \forall t \in T, \forall S \text{ is a subtour},$$

$$0 \leq x_{it} \leq U_i \sum_{j \in V, j \neq i} y_{ij}, \forall i \in V', \forall t \in T,$$

$$x_{it} + \sum_{i \in V'} x_{it}' = 0, \forall t \in T,$$

$$\sum_{i \in V'} x_{it}' \leq Q, \forall t \in T,$$

$$I_i^1 = I_i^{t-1} + x_{it} - r_{it}' \forall i \in V, \forall t \in T,$$

$$I_i^{t+1} + x_{it}' \leq U_i, \forall i \in V', \forall t \in T,$$

$$I_0^{t+1} + r_{it}' \leq U_0, \forall t \in T,$$

$$0 \leq I_i^t \leq U_i, \forall i \in V, \forall t \in T,$$

$$x_{it} \in R, \forall i, j \in V, i \neq j, \forall t \in T.$$}

Objective (1) minimizes the sum of the inventory costs and routing costs. Constraints (2) ensure that each vertex can be visited at most once in each period, and its in-degree equals to the out-degree. Constraints (3) are for subtour elimination where a subtour is a cycle which does not start from the depot [Dantzig et al., 1954]. Constraints (4) indicate that there can be positive delivery quantity for each client in each period iff the client is visited, and the delivery quantity must not exceed the capacity of the client. Constraints (5) require that in each period, the quantity loaded at the depot must match the total quantity delivered to all clients. Constraints (6) guarantee that the vehicle will never be overloaded. Constraints (7) are the inventory conservation equalities. Constraints (8) and (9) impose the capacity of each vertex. Constraints (10) forbid the shortage at customers. Constraints (11) present the domains of the decision variables.

### 3 Two-Stage Matheuristic Algorithm

The IRP integrates both inventory management and vehicle routing, thus involves both continuous and discrete optimizations which are usually solved by different methodologies. As a hybrid algorithm which combines the advantages of mathematical programming and metaheuristics, the matheuristic algorithm is a promising approach for such kind of problems. Meanwhile, due to the complexity of the IRP, it is very challenging to optimize the whole problem directly. Therefore, the proposed TSMH algorithm consists of two stages whose search granularities vary from high-level structure to low-level details. Specifically, the first stage generates the initial solution by row generation approach with additional repairing procedure (Section 3.1). The second stage adopts a solution-based tabu search procedure to implement the fine-grained optimization (Section 3.2).

In order to find out a proper overall structure of the delivery schedule, the proposed algorithm relaxes the MIP model for the IRP to obtain an easier subproblem which optimizes the timing and the quantity of the deliveries while considering the routing as accurate as possible. Specifically, due to the exponential complexity of the subtour elimination constraints (3), relaxing them will greatly reduce the intractability of the induced subproblem. However, once a solution for the relaxed model is found, the subtour elimination constraints will be lazily added to claim its infeasibility. Although the relaxed solutions may not form connected tours, we can decide whether a vertex $i$ is visited or not in period $t$ by examining $Z_{ij}^t = \sum_{j \in V, j \neq i} y_{ij}$. Furthermore, in each period, when the vertices to visit (black nodes in Figure 1) are fixed, repairing the route is equivalent to solving the traveling salesman problem (TSP) on the subgraph composed of the visited vertices. The TSMH algorithm will repeat the above solve-tighten-repair procedure until the time limit is reached.

When the quality of the best found solution gradually converges in the first stage, the TSMH algorithm moves on to the second stage. This stage employs a solution-based iterated tabu search algorithm started from the best solution found so far to carry out a fine-grained optimization. This tabu search algorithm perturbs the current best solution when it fails to improve the best solution found for a number of consecutive iterations. Finally, the TSMH stops after the time limit is reached and returns the best solution found so far.

#### 3.1 Stage 1: Structural Optimization by Relax-and-Repair Method

We consider a relaxed IRP (RIRP) model which consists of Eqs. (1)-(2) and (4)-(11) in the first stage. As described in Algorithm 1, the TSMH solves the RIRP model to obtain relaxed solutions (line 3). In these solutions, the inventory-related constraints are respected but there may be subtours disconnected to the depot. Hence, for each period, we drop the clients with zero delivery quantity or $Z_{ij}^t = 0$, and solve the TSP on the remaining subgraph using Lin-Kernighan heuristic [Lin and Kernighan, 1973] (line 4). This repairing procedure is able to efficiently produce feasible solutions while preserving the overall structure of the relaxed solutions. If the cost of the repaired solution is less than the best solu-
Algorithm 1 Structural Optimization

Input: Instance $I$; 
Output: Best found solution $S_{best}$

1: $S_{best} \leftarrow \emptyset$
2: repeat
3: Solve RIRP model to get a relaxed solution $S_{rel}$
4: Repair the tours in $S_{rel}$ to get a feasible solution $S_{cur}$
5: if $f(S_{cur}) < f(S_{best})$ then
6: $S_{best} \leftarrow S_{cur}$
7: end if
8: Add a constraint which eliminate the minimum sub-tour (if there exists) to RIRP model
9: until time limit is reached or no sub-tour in $S_{rel}$

tion found so far, the best solution will be updated (lines 5-7). 
Next, one of the violated subtour elimination constraints involving the least number of edges will be added to the RIRP model as lazy constraints (line 11). After adding the lazy constraints, TSMH resumes the optimization and iteratively executes the above procedure until the time limit is reached (line 2) or a feasible solution is found (lines 8-10). This stage is able to produce good initial solutions, and it may directly obtain the optimal solutions on some small instances.

3.2 Stage 2: Detailed Refinement by Solution-based Tabu Search

This stage adopts a solution-based iterated tabu search [Lai et al., 2018] to perform a fine-grained search starting from the best solution obtained in the previous stage. Similar to typical tabu search algorithms, the proposed solution-based tabu search iteratively evaluates the neighborhood of the current solution and performs the best non-tabu neighborhood move. If it is unable to improve the best solution found for a given number of iterations, a perturbation procedure is launched. The above procedures are executed repeatedly until the time limit is reached. However, unlike the traditional attribute-based tabu search, the proposed TSMH algorithm employs a solution-based tabu strategy which records the complete information of each evaluated solution. If we regard the limited characteristics in the attribute-based tabu list as directions or landmarks in the solution space, the ones in the solution-based tabu list will be the exact coordinates. Next, we will present comprehensive descriptions for the essential ingredients of the solution-based tabu search subroutine.

Neighborhood Structure and Evaluation

We define four neighborhood structures to optimize the delivery schedule. Unlike the classical ones for routing problems, these neighborhoods manipulate the timing of visits to the clients ($Z^t_i$) instead of reordering the routes. At each iteration of the tabu search, the current solution $S$ is replaced with its neighboring solution by performing the best move $M$ selected from all the following neighborhoods.

- Addition operation $M_a(t, i)$. Visit unvisited client $i$ in period $t$. There are $O(pn)$ addition operations.
- Removal operation $M_r(t, i)$. Cancel the existing visit to client $i$ in period $t$. There are $O(pn)$ removal operations.
- Move operation $M_m(t_1, t_2, i) = M_a(t_1, i) + M_r(t_2, i)$. Visit client $i$ in period $t_1$ if it is not visited, and simultaneously cancel the existing visit to the same client in period $t_2$. There are $O(p^2n)$ move operations.
- Swap operation $M_s(t_1, t_2, i, j) = M_m(t_1, t_2, i) + M_m(t_1, t_2, j)$. Exchange the visits to a pair of clients in a pair of periods where $Z^t_{i_1} = Z^t_{j_1} = 1$, $Z^t_{i_2} = Z^t_{j_2} = 0$. There are $O(p^2n^2)$ swap operations.

The neighborhood evaluation is the bottleneck of the tabu search algorithm, since it evaluates a lot of moves but performs only one of them at each iteration. In order to improve the search efficiency, our TSMH algorithm adopts a filtering technique and only applies exact evaluations on promising moves. The proposed filtering technique evaluates the entire neighborhood in an incremental and approximate way, and then it abandons the moves with poor approximate objective improvements. Specifically, the approximate objective function only considers the routing costs, and the objective improvements $\Delta M$ of move $M$ are calculated as Eqs. (12).

\[
\Delta M_a(t, i) = \min_{(t_1, t_2) \in \text{tour}} \{C_{ii1} + C_{ii2} - C_{ii12}\},
\]
\[
\Delta M_r(t, i) = C_{ii12} - C_{ii1} - C_{ii2},
\]
\[
\Delta M_m(t_1, t_2, i) = \Delta M_a(t_1, i) + \Delta M_r(t_2, i),
\]
\[
\Delta M_s(t_1, t_2, i, j) = \Delta M_m(t_1, t_2, i) + \Delta M_m(t_1, t_2, j).
\]

Eqs. (12) are incremental evaluations which assume that only the predecessors $t_1$ and successors $t_2$ of the added or removed visits will be affected in the original tour. For example, when removing client $i$ from tour ($..., t_1, i, t_2, ...$) in period $t$, it assumes that the optimal tour for the resulting vertex combination is ($..., t_1, t_2, ...$). Based on Eqs. (12), we can sort the neighborhood moves by $\Delta M$ values in an ascending order, and only take the top-$m$ elite moves into further consideration. Thus, the filtering technique can efficiently identify the non-promising moves, at the cost of possible inaccuracy.

Next, the proposed TSMH figures out the exact holding costs of the elite solutions. Since the visiting states are the only band to connect the inventory and routing subproblems as Eqs. (4) impose, the holding and routing costs are independent to each other once the visiting states $Z^t_i$ are fixed, which is true at this moment. So, we just leave out the routing part and minimize the inventory holding costs $f_h(S)$ by solving the LP model composed of (5)-(11) and (13)-(14). Note that we treat the move as the worst if its LP model is infeasible.

\[
\min f_h(S) = \sum_{t \in T \cup \{0\}} \sum_{i \in V} f_i H_i^t
\]
\[
0 \leq x_i^t \leq U_i Z_i^t, \forall i \in V^t, \forall t \in T
\]

Finally, we make the move with the smallest $f_h(S \ominus M) + \Delta M$ value and re-optimize the routes in each affected period using Lin-Kernighan heuristic. As discussed in Section 3.1, the route optimization in period $t$ is equivalent to solving the TSP in a subgraph each of whose vertex $i$ satisfies $Z_i^t = 1$.

Solution-based Tabu Strategy

In order to jump out of the local optima during the neighborhood search, we integrate a solution-based tabu strategy into the TSMH algorithm. Comparing to its classical
attribute-based counterpart, the solution-based tabu strategy records more complete information of each evaluated solution. Therefore, we can omit the tabu tenure parameter and prohibit visiting the evaluated solutions forever. This feature overcomes the drawbacks of the attribute-based tabu search that inappropriate parameter settings may falsely forbid some promising moves or accept moving back to visited solutions. For efficiency consideration, we only store the hash values of the evaluated solutions instead of the entire solutions. We define a hash function \( h : S \rightarrow \{0, 1, 2, \ldots, K-1\} \) to map the solution vector \( S \) into non-negative integers ranging from 0 to \( K-1 \). Then the tabu list is implemented by a Boolean vector \( B \) containing \( K \) entries, where \( B_k \) is true indicates that the solution with the hash value \( h(S) = k \) is in the tabu state. Due to the possibility of hash collision, we employ multiple hash functions \( h^j \) and corresponding tabu lists \( B^j \) for \( j = 1, 2, \ldots, J \) to improve the robustness of the solution-based tabu strategy. In the proposed TSMH algorithm, we take \( J = 3 \), and a solution \( S \) is in tabu state iff \( \bigwedge_{j=1}^{J} B^j_{h^j(S)} \) is true. Correspondingly, we need to set the above three entries to true for each evaluated solution. We define hash function \( h^j \) as follows, where \( \gamma_j \) is a parameter.

\[
h^j(S) = \sum_{i \in I} \sum_{v \in V^r} \left( (tn + i) \gamma^j \right) Z^i_v \mod K, \forall j \in J \quad (15)
\]

**Perturbation Operator**

When the tabu search fails to improve the current best solution for consecutive \( \alpha \) iterations, TSMH utilizes a perturbation operator to escape from the local optima. The perturbation starts from the best solution found so far, and successively performs \( \pi_a \) addition, \( \pi_r \) removal, and \( \pi_m \) move operations. Each operation is selected from all neighborhood moves in the corresponding neighborhood from a uniform distribution. In addition, an operation that neutralizes the effect of another one will not be selected. If the resulting solution is infeasible or in tabu state, the perturbation will restart until a proper solution is obtained.

### 4 Computational Experiment and Analysis

#### 4.1 Experimental Protocols

The proposed TSMH algorithm is evaluated on two sets of most commonly used datasets, which are 160 small instances and 60 large instances proposed by Archetti et al. [2007] and 60 large ones proposed by Archetti et al. [2012], respectively. In the small-scale dataset, there are 100 three-period instances with 5, 10, ..., 50 clients and 60 six-period ones whose numbers of clients are 5, 10, ..., 30. The large instances consist of 6 periods and 50, 100, 200 clients. Each configuration can be further divided into two groups according to the distributions of their unit inventory costs, which are \([0.01, 0.05]\) (low cost) and \([0.1, 0.5]\) (high cost), respectively. In addition, the traveling cost between each pair of vertices \( i, j \) is calculated by rounding the Euclidean distance between them.

Our TSMH algorithm is developed in C++ and tested on Intel Xeon E5-2698 v3 2.30GHz CPU. The MIP models are solved by Gurobi 8.0 [Gurobi Optimization, 2018], and the TSP subproblems are solved by LKH3 [Helsgaun, 2017]. We perform 20 independent runs for each instance under a one-hour time limit and record the best results. We also extend the time limit to 3 hours to investigate the potential of the proposed TSMH algorithm. The time limit for the first stage is 1/6 of the total time limit. The values of other parameters for TSMH are listed in Table 1.

#### 4.2 Computational Results

In order to justify the effectiveness and efficiency of the proposed algorithm, we compare TSMH with the best known exact algorithm CL-BC [Coelho and Laporte, 2013; Coelho and Laporte, 2014] and heuristic algorithm HAIR [Archetti et al., 2012]. The computational platforms for CL-BC and HAIR are Xeon 2.66GHz and Intel Dual Core 1.86GHz CPU, respectively. By assuming the hardware performance is proportional to the CPU frequency, we will scale down the computational time of HAIR to 80% for a fair comparison.

The overall computational results of each group of instances are shown in Table 2. Column #best indicates the number of instances where the corresponding algorithm can match the best known objective values. Column Gap reports the average gap between the best result obtained by each algorithm and the best known one. Column Time gives the normalized average computational time in seconds for obtaining the reported results. From Table 2 we can observe that, TSMH is a robust algorithm whose average gap on all 220 instances is 7 and 150 times smaller than HAIR and CL-BC. Moreover, the proposed algorithm is highly scalable that it obtains most best known solutions on the two sets of large instances. In addition, TSMH is also very efficient as the average computational time is 3 times shorter than CL-BC.

Tables 3 and 4 present the detailed results and comparisons on the 60 large instances. Column Instance gives the reformatted name of each large instance according to their characteristics. For example, 56c2n50.4 indicates the fourth instance where there are 6 periods and 50 customers and the scale of the unit inventory costs are \(10^{-2}\) (low cost). The next five columns report the objective values obtained by

---

**Table 1: Parameter settings.**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>( m )</td>
<td>Maximal number of neighborhood moves</td>
<td>2π/( \sqrt{n} )</td>
</tr>
<tr>
<td>( \alpha )</td>
<td>Maximal number of non-improving moves</td>
<td>25</td>
</tr>
<tr>
<td>( K )</td>
<td>Range of hash values</td>
<td>( 10^6 )</td>
</tr>
<tr>
<td>( \gamma_1 ), ( \gamma_2 ), ( \gamma_3 )</td>
<td>Parameters for the hash functions</td>
<td>( 1.8, 2.4, 3.0 )</td>
</tr>
<tr>
<td>( \pi_a )</td>
<td>Number of addition operations in perturbation</td>
<td>( {2, 3} )</td>
</tr>
<tr>
<td>( \pi_r )</td>
<td>Number of removal operations in perturbation</td>
<td>( {1, 2} )</td>
</tr>
<tr>
<td>( \pi_m )</td>
<td>Number of move operations in perturbation</td>
<td>( {4, 5, 6} )</td>
</tr>
</tbody>
</table>

**Table 2: Average computational results on all groups of instances.**

<table>
<thead>
<tr>
<th>Instance group</th>
<th>CL-BC</th>
<th>HAIR</th>
<th>TSMH</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scale</td>
<td>Cost Count</td>
<td>#best Gap (%)</td>
<td>Time</td>
</tr>
<tr>
<td>Small 3 low</td>
<td>50</td>
<td>50</td>
<td>0.000</td>
</tr>
<tr>
<td>Small 3 high</td>
<td>50</td>
<td>50</td>
<td>0.000</td>
</tr>
<tr>
<td>Small 6 low</td>
<td>30</td>
<td>30</td>
<td>0.000</td>
</tr>
<tr>
<td>Small 6 high</td>
<td>30</td>
<td>30</td>
<td>0.000</td>
</tr>
<tr>
<td>Large 6 low</td>
<td>30</td>
<td>13</td>
<td>14.900</td>
</tr>
<tr>
<td>Large 6 high</td>
<td>30</td>
<td>13</td>
<td>7.230</td>
</tr>
<tr>
<td>Sum/average</td>
<td>220</td>
<td>186</td>
<td>3.018</td>
</tr>
</tbody>
</table>
Table 3: Large instances with high inventory costs.

<table>
<thead>
<tr>
<th>Instance</th>
<th>CL-BC</th>
<th>HAIR</th>
<th>TSMH</th>
</tr>
</thead>
<tbody>
<tr>
<td>p6c1n50.1</td>
<td>9966.14</td>
<td>9974.38</td>
<td>9974.38</td>
</tr>
<tr>
<td>p6c1n50.2</td>
<td>10632.00</td>
<td>10632.24</td>
<td>10632.24</td>
</tr>
<tr>
<td>p6c1n50.3</td>
<td>10510.70</td>
<td>10548.98</td>
<td>10548.98</td>
</tr>
<tr>
<td>p6c1n50.4</td>
<td>10513.00</td>
<td>10555.38</td>
<td>10555.38</td>
</tr>
<tr>
<td>p6c1n50.5</td>
<td>10113.00</td>
<td>10137.40</td>
<td>10137.40</td>
</tr>
<tr>
<td>p6c1n50.6</td>
<td>10148.02</td>
<td>10166.10</td>
<td>10166.10</td>
</tr>
<tr>
<td>p6c1n50.7</td>
<td>9982.20</td>
<td>10012.68</td>
<td>10012.68</td>
</tr>
<tr>
<td>p6c1n50.8</td>
<td>10009.90</td>
<td>10052.78</td>
<td>10052.78</td>
</tr>
<tr>
<td>p6c1n50.9</td>
<td>9656.20</td>
<td>9727.74</td>
<td>9727.74</td>
</tr>
</tbody>
</table>

Table 4: Large instances with low inventory costs.

<table>
<thead>
<tr>
<th>Instance</th>
<th>CL-BC</th>
<th>HAIR</th>
<th>TSMH</th>
</tr>
</thead>
<tbody>
<tr>
<td>p6c1n50.1</td>
<td>2034.80</td>
<td>2034.80</td>
<td>2034.80</td>
</tr>
<tr>
<td>p6c1n50.2</td>
<td>1769.71</td>
<td>1769.71</td>
<td>1769.71</td>
</tr>
<tr>
<td>p6c1n50.3</td>
<td>10626.50</td>
<td>10626.50</td>
<td>10626.50</td>
</tr>
<tr>
<td>p6c1n50.4</td>
<td>10497.20</td>
<td>10497.20</td>
<td>10497.20</td>
</tr>
<tr>
<td>p6c1n50.5</td>
<td>110709.97</td>
<td>110709.97</td>
<td>110709.97</td>
</tr>
<tr>
<td>p6c1n50.6</td>
<td>7616.49</td>
<td>7616.49</td>
<td>7616.49</td>
</tr>
<tr>
<td>p6c1n50.7</td>
<td>3391.99</td>
<td>3391.99</td>
<td>3391.99</td>
</tr>
<tr>
<td>p6c1n50.8</td>
<td>3376.60</td>
<td>3376.60</td>
<td>3376.60</td>
</tr>
<tr>
<td>p6c1n50.9</td>
<td>3572.83</td>
<td>3572.83</td>
<td>3572.83</td>
</tr>
</tbody>
</table>

Research partially supported by Alkaid lab of Huawei Cloud.
References


