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

The rise of e-business brings great opportunities for the supply chain industry, along with challenges due to the growing demands for higher throughput and lower latency. Fortunately, the developments of Internet of Things and statistical learning makes it possible to collect sufficient data from every link in the value chain and provide constructive analysis and predictions. By utilizing these data, the related companies are able to reduce their operation expenses, increases the profits, improve the efficiency, and bring better user experiences. Thus, more and more companies start to pay attentions to systematic optimization for the whole supply chain.

[*]Corresponding author

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.

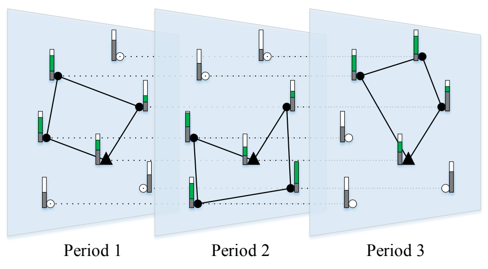

Figure 1: Example of an IRP instance and its solution. The triangle is the depot. The black nodes represent the visited clients in each period and the white nodes are the unvisited ones. The gray, green, and white bars stand for remaining inventory levels, delivered quantities, and free capacities, respectively. The lines are the delivery routes.

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 subproblems, 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.

2 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

[1]http://www.roadef.org/challenge/2016/en/sujet.php

[2]http://dimacs.rutgers.edu/events/details?eID=1090

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_0^t units of products and client $i \in V'$ consumes r_i^t 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_i^t to be the quantity of products delivered to vertex $i \in V$ in period $t \in T$. Obviously, x_0^t is always non-positive since the vehicle can only load instead of delivering products at depot. We use Boolean variable y_{ij}^t 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.

$$\min \quad \sum_{t \in T \cup \{0\}} \sum_{i \in V} H_i I_i^t + \sum_{t \in T} \sum_{i \in V} \sum_{j \in V, j \neq i} C_{ij} y_{ij}^t, \tag{1}$$

$$\text{s.t.} \quad \sum_{j \in V, j \neq i} y_{ij}^t = \sum_{j \in V, j \neq i} y_{ji}^t \leq 1, \forall i \in V, \forall t \in T, \tag{2}$$

$$\sum_{i,j \in S} y_{ij}^t \leq |S| - 1, \forall t \in T, \forall S \text{ is a subtour}, \tag{3}$$

$$0 \leq x_i^t \leq U_i \sum_{j \in V, j \neq i} y_{ij}^t, \forall i \in V', \forall t \in T, \tag{4}$$

$$x_0^t + \sum_{i \in V'} x_i^t = 0, \forall t \in T, \tag{5}$$

$$\sum_{i \in V'} x_i^t \leq Q, \forall t \in T, \tag{6}$$

$$I_i^t = I_i^{t-1} + x_i^t - r_i^t, \forall i \in V, \forall t \in T, \tag{7}$$

$$I_i^{t-1} + x_i^t \leq U_i, \forall i \in V', \forall t \in T, \tag{8}$$

$$I_0^{t-1} + r_0^t \leq U_0, \forall t \in T, \tag{9}$$

$$0 \leq I_i^t \leq U_i, \forall i \in V, \forall t \in T, \tag{10}$$

$$x_i^t \in R, \; y_{ij}^t \in \{0, 1\}, \forall i, j \in V, i \neq j, \forall t \in T. \tag{11}$$

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 $Z_i^t = \sum_{j \in V, j \neq i} y_{ij}^t$. 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 $Z_i^t = 0$, 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 StructuralOptimization

```
Input: Instance $I$;
Output: Best found solution $S_{best}$
1: $S_{best} \leftarrow \varnothing$
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 subtour (if there exists) to $RIRP$ model
9: until time limit is reached or no subtour 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_i^t) 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 $M_s(t_i,t_j,i,j) = M_m(t_j,t_i,i) + M_m(t_i,t_j,j)$. Exchange the visits to a pair of clients in a pair of periods where $Z_i^{t_i} = Z_j^{t_j} = 1, Z_i^{t_j} = Z_j^{t_i} = 0$. 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 ΔM of move M are calculated as Eqs. (12).

$$\begin{aligned}
&\Delta M_a(t,i) = \min_{(i_1,i_2)\in \text{tour}} \{C_{i_1 i} + C_{i i_2} - C_{i_1 i_2}\},\\
&\Delta M_r(t,i) = C_{i_1 i_2} - C_{i_1 i} - C_{i i_2},\\
&\Delta M_m(t_1,t_2,i) = \Delta M_a(t_1,i) + \Delta M_r(t_2,i),\\
&\Delta M_s(t_i,t_j,i,j) = \Delta M_m(t_j,t_i,i) + \Delta M_m(t_i,t_j,j).
\end{aligned} \tag{12}$$

Eqs. (12) are incremental evaluations which assume that only the predecessors i_1 and successors i_2 of the added or removed visits will be affected in the original tour. For example, when removing client i from tour $(..., i_1, i, i_2, ...)$ in period t, it assumes that the optimal tour for the resulting vertex combination is $(..., i_1, i_2, ...)$. Based on Eqs. (12), we can sort the neighborhood moves by ΔM 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_i^t 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.

$$\min f_h(S) = \sum_{t\in T\cup\{0\}} \sum_{i\in V} H_i I_i^t \tag{13}$$

$$0 \le x_i^t \le U_i Z_i^t, \forall i \in V', \forall t \in T \tag{14}$$

Finally, we make the move with the smallest $f_h(S \oplus M) + \Delta M$ 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 $h : S \to \{0, 1, 2, ..., K-1\}$ 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 $j = 1, 2, ..., J$ 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 γ_j is a parameter.

$$h^j(S) = \sum_{t \in T} \sum_{i \in V'} \lfloor (tn+i)^{\gamma_j} \rfloor Z_i^t \mod K, \forall j \in J \quad (15)$$

Perturbation Operator

When the tabu search fails to improve the current best solution for consecutive α 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 π_a addition, π_r removal, and π_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 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

Parameter	Description	Value
m	Maximal number of elite neighborhood moves	$2p\sqrt{n}$
α	Maximal number of non-improving moves	25
K	Range of hash values	10^8
$\gamma_1, \gamma_2, \gamma_3$	Parameters for the hash functions	1.8, 2.4, 3.0
π_a	Number of addition operations in perturbation	$\{2, 3\}$
π_r	Number of removal operations in perturbation	$\{1, 2\}$
π_m	Number of move operations in perturbation	$\{4, 5, 6\}$

Table 1: Parameter settings.

Instance group				CL-BC			HAIR			TSMH		
Scale	p	Cost	Count	#best	Gap (%)	Time	#best	Gap (%)	Time	#best	Gap (%)	Time
Small	3	low	50	50	0.000	10	49	0.000	262	49	0.001	18
Small	3	high	50	50	0.000	10	48	0.001	274	49	0.000	37
Small	6	low	30	30	0.000	37	12	0.121	571	27	0.019	330
Small	6	high	30	30	0.000	29	11	0.165	490	27	0.005	435
Large	6	low	30	13	14.900	14400	2	0.440	-	18	0.070	4567
Large	6	high	30	13	7.230	14400	1	0.350	-	22	0.040	3642
Sum/average			220	186	3.018	3942	123	0.147	-	192	0.019	1239

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

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 are 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, p6c2n50.4 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

Instance	CL-BC	HAIR		TSMH		
	4 hours	1 hour	>1 hour	1 hour	3 hours	Time (s)
p6c1n50.1	**30189.40**	30225.36	30225.36	30189.60	30189.60	2208.20
p6c1n50.2	**29790.00**	29856.26	29856.26	**29790.00**	**29790.00**	2034.80
p6c1n50.3	**29790.90**	29904.15	29904.15	**29790.90**	**29790.90**	1209.52
p6c1n50.4	**31518.30**	31677.87	31677.87	31542.30	31542.30	1978.24
p6c1n50.5	**29240.40**	29400.33	29400.33	29241.60	29241.60	2029.23
p6c1n50.6	**31903.10**	32195.51	31946.33	31980.00	**31903.10**	7616.49
p6c1n50.7	**29734.50**	29768.03	29768.03	29736.80	29736.80	2458.45
p6c1n50.8	**25954.20**	26521.96	26521.96	**25954.20**	**25954.20**	2165.80
p6c1n50.9	**30192.90**	30283.90	30283.90	**30192.90**	**30192.90**	1769.71
p6c1n50.10	**31338.20**	31397.84	31397.84	**31338.20**	**31338.20**	1696.02
p6c1n100.1	**57459.20**	57721.23	57721.23	57498.30	57498.30	3139.07
p6c1n100.2	53510.10	53432.80	53432.80	53452.00	**53412.70**	9907.20
p6c1n100.3	58505.10	58598.93	58598.93	**58502.60**	**58502.60**	3369.19
p6c1n100.4	**51554.20**	52030.59	52030.59	51739.20	51739.20	3157.25
p6c1n100.5	**57976.50**	58258.92	58258.92	58144.00	58144.00	2653.80
p6c1n100.6	55087.80	55280.01	55280.01	**55074.00**	**55074.00**	3343.64
p6c1n100.7	56076.90	56398.19	56398.19	**56074.00**	**56074.00**	3391.99
p6c1n100.8	56057.10	55722.32	55384.47	**55060.95**	**55060.95**	3298.24
p6c1n100.9	59425.90	58729.94	58729.94	**58496.60**	**58496.60**	3512.04
p6c1n100.10	56588.30	56644.22	56644.22	**56448.10**	**56448.10**	2590.43
p6c1n200.1	136337.00	110790.39	110790.39	**110709.97**	**110709.97**	2762.99
p6c1n200.2	141543.00	112424.46	112401.98	112464.00	**112385.00**	8292.39
p6c1n200.3	123147.00	108119.61	108119.61	**107865.00**	**107865.00**	3371.67
p6c1n200.4	129615.00	109309.48	109309.48	**108852.00**	**108852.00**	3510.55
p6c1n200.5	126552.00	109104.89	109083.07	**109021.00**	**109021.00**	3205.40
p6c1n200.6	136513.00	**109071.20**	**109071.20**	109396.00	109348.00	10492.80
p6c1n200.7	111186.00	97749.52	97749.52	**97720.60**	**97720.60**	3572.83
p6c1n200.8	115946.00	102194.63	102194.63	**102066.00**	**102066.00**	3542.09
p6c1n200.9	136819.00	104877.49	104877.49	**104723.00**	**104723.00**	3475.74
p6c1n200.10	142796.00	109066.98	109045.17	**108765.00**	**108765.00**	3496.09
#best	13/30	1/30	1/30	19/30	22/30	
Average	72078.23	64558.57	64536.80	64394.29	64386.19	3641.73

Table 3: Large instances with high inventory costs.

Instance	CL-BC	HAIR		TSMH		
	4 hours	1 hour	>1 hour	1 hour	3 hours	Time (s)
p6c2n50.1	**9966.14**	9974.38	9974.38	9971.71	9971.71	2241.31
p6c2n50.2	**10632.00**	10632.24	10632.24	10633.80	**10632.00**	4506.10
p6c2n50.3	**10510.70**	10548.98	10548.98	10511.80	10511.80	1884.15
p6c2n50.4	**10513.40**	10555.38	10555.38	10517.90	10517.90	1362.32
p6c2n50.5	**10113.00**	10137.40	10137.40	10113.80	10113.80	2265.85
p6c2n50.6	**10148.02**	10166.10	10166.10	10151.20	10151.20	2306.24
p6c2n50.7	**9982.20**	10012.68	10012.68	9984.53	9984.53	2215.91
p6c2n50.8	**10299.10**	10547.97	10547.97	**10299.10**	**10299.10**	2183.55
p6c2n50.9	**10009.90**	10052.78	10052.78	10022.60	10022.60	1809.06
p6c2n50.10	**9659.20**	9727.74	9727.74	**9659.20**	**9659.20**	1866.08
p6c2n100.1	**15649.30**	15784.02	15784.02	15684.80	15684.80	3204.62
p6c2n100.2	14697.30	14754.10	14754.10	**14632.60**	**14632.60**	3137.05
p6c2n100.3	16154.60	15649.44	15649.44	**15630.90**	**15630.90**	2271.17
p6c2n100.4	**14644.30**	14755.76	14755.76	14737.30	14737.30	3514.99
p6c2n100.5	**15234.80**	15412.83	15412.83	15314.50	15314.50	3197.03
p6c2n100.6	15769.40	15327.08	15327.08	**15252.00**	**15252.00**	3496.13
p6c2n100.7	15537.70	15468.98	15468.98	**15374.60**	**15374.60**	3585.08
p6c2n100.8	15279.20	15078.16	15078.16	**14996.70**	**14996.70**	3604.63
p6c2n100.9	17189.60	15628.66	15628.66	**15584.20**	**15584.20**	3376.60
p6c2n100.10	16145.00	15495.87	15495.87	**15467.30**	**15467.30**	3591.65
p6c2n200.1	32683.30	24353.40	24353.40	24413.90	**24285.20**	9597.22
p6c2n200.2	34033.40	24610.96	24576.55	24591.10	**24484.60**	10497.20
p6c2n200.3	33317.20	23861.65	23861.65	23941.00	**23793.80**	10626.50
p6c2n200.4	34004.10	24329.66	24232.73	24286.80	**24175.40**	8334.02
p6c2n200.5	35486.90	**24193.98**	**24193.98**	24337.90	24223.60	8339.13
p6c2n200.6	33359.50	23651.81	23651.81	**23633.40**	**23633.40**	3410.45
p6c2n200.7	32773.90	23527.12	23527.12	**23388.60**	**23388.60**	2986.97
p6c2n200.8	33488.70	23311.73	**23230.42**	23341.90	23322.80	9182.80
p6c2n200.9	35172.60	23846.19	23846.19	24006.90	**23840.80**	8802.92
p6c2n200.10	34871.50	23609.34	23609.34	23653.40	**23516.40**	9627.85
#best	13/30	1/30	2/30	11/30	18/30	
Average	19910.87	16500.21	16493.12	16471.18	16440.11	4567.49

Table 4: Large instances with low inventory costs.

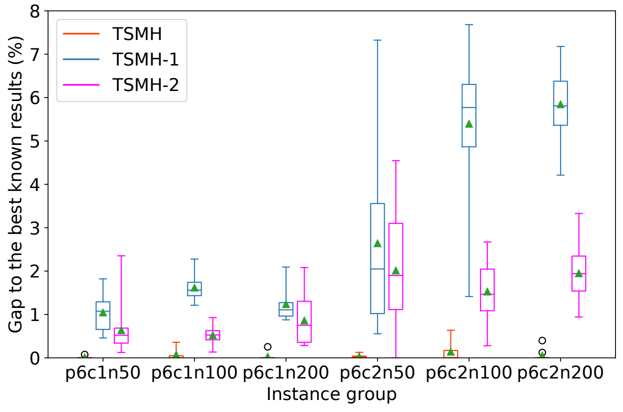

Figure 2: Comparing TSMH to its two simplified versions.

each algorithm under the specified time limits. Column Time presents the computational time taken by TSMH to obtain its best results. The numbers in bold indicate the best known results. As we can see, although CL-BC obtains the best results on all 20 instances with 50 clients under a four-hour time limit, TSMH shows its advantage on larger instances. As a result, TSMH obtains the best known results on 40 instances, while CL-BC only found 26 ones. Regarding the one-hour time limit, TSMH outperforms HAIR in terms of both the number of best solutions and the average objective values. In sum, the proposed TSMH algorithm is competitive comparing to the state-of-the-art algorithms in the literature.

4.3 Effectiveness of Combining Two Stages

To demonstrate the importance of each stage, we conduct extensive experiments to compare the complete TSMH algorithm with its two simplified versions TSMH-1 and TSMH-2 under one-hour time limit. TSMH-1 only includes the structural optimization stage. TSMH-2 generates the initial solution by solving the RIRP model without adding lazy constraints, and incorporates the detailed refinement stage based on the tabu search algorithm. Figure 2 plots the distribution of the solution qualities obtained by these three versions. We can observe that it fails to match any best known result if there is only a single stage. In addition, the solution-based tabu search obtains relatively better results comparing to the relax-and-repair method. These results highlight the merit of the hybridization of different methodologies.

5 Conclusion

We proposed a two-stage matheuristic algorithm which includes a structural optimization stage and a detailed refinement stage. Tested on 220 most commonly used instances, the TSMH algorithm improves the best known results on 40 out of 60 large instances. In addition, extensive experiments show that the combination of the two stages is essential to the performance of TSMH.

Acknowledgments

Research partially supported by Alkaid lab of Huawei Cloud.

References

[Adulyasak *et al.*, 2013] Yossiri Adulyasak, Jean-François Cordeau, and Raf Jans. Formulations and branch-and-cut algorithms for multivehicle production and inventory routing problems. *INFORMS Journal on Computing*, 26(1):103–120, 2013.

[Aksen *et al.*, 2014] Deniz Aksen, Onur Kaya, F Sibel Salman, and Özge Tüncel. An adaptive large neighborhood search algorithm for a selective and periodic inventory routing problem. *European Journal of Operational Research*, 239(2):413–426, 2014.

[Alvarez *et al.*, 2018] Aldair Alvarez, Pedro Munari, and Reinaldo Morabito. Iterated local search and simulated annealing algorithms for the inventory routing problem. *International Transactions in Operational Research*, 25(6):1785–1809, 2018.

[Archetti *et al.*, 2007] Claudia Archetti, Luca Bertazzi, Gilbert Laporte, and Maria Grazia Speranza. A branch-and-cut algorithm for a vendor-managed inventory-routing problem. *Transportation Science*, 41(3):382–391, 2007.

[Archetti *et al.*, 2012] Claudia Archetti, Luca Bertazzi, Alain Hertz, and M Grazia Speranza. A hybrid heuristic for an inventory routing problem. *INFORMS Journal on Computing*, 24(1):101–116, 2012.

[Archetti *et al.*, 2017] Claudia Archetti, Natashia Boland, and M Grazia Speranza. A matheuristic for the multivehicle inventory routing problem. *INFORMS Journal on Computing*, 29(3):377–387, 2017.

[Avella *et al.*, 2017] Pasquale Avella, Maurizio Boccia, and Laurence A Wolsey. Single-period cutting planes for inventory routing problems. *Transportation Science*, 52(3):497–508, 2017.

[Becraft *et al.*, 2012] Warren R Becraft, Dom Kalasih, Stephen Brooks, et al. Replenishment planning and secondary petroleum distribution optimisation at z energy. In *Chemeca 2012: Quality of life through chemical engineering*. Engineers Australia, 2012.

[Bertazzi *et al.*, 2002] Luca Bertazzi, Giuseppe Paletta, and M Grazia Speranza. Deterministic order-up-to level policies in an inventory routing problem. *Transportation Science*, 36(1):119–132, 2002.

[Bertazzi *et al.*, 2019] Luca Bertazzi, Leandro C Coelho, Annarita De Maio, and Demetrio Laganà. A matheuristic algorithm for the multi-depot inventory routing problem. *Transportation Research Part E: Logistics and Transportation Review*, 122:524–544, 2019.

[Campbell and Savelsbergh, 2004] Ann Melissa Campbell and Martin WP Savelsbergh. A decomposition approach for the inventory-routing problem. *Transportation Science*, 38(4):488–502, 2004.

[Chitsaz *et al.*, 2019] Masoud Chitsaz, Jean-François Cordeau, and Raf Jans. A unified decomposition matheuristic for assembly, production, and inventory routing. *INFORMS Journal on Computing*, 31(1):134–152, 2019.

[Coelho and Laporte, 2013] Leandro C Coelho and Gilbert Laporte. The exact solution of several classes of inventory-routing problems. *Computers & Operations Research*, 40(2):558–565, 2013.

[Coelho and Laporte, 2014] Leandro C Coelho and Gilbert Laporte. Improved solutions for inventory-routing problems through valid inequalities and input ordering. *International Journal of Production Economics*, 155:391–397, 2014.

[Coelho *et al.*, 2012] Leandro C Coelho, Jean-François Cordeau, and Gilbert Laporte. Consistency in multi-vehicle inventory-routing. *Transportation Research Part C: Emerging Technologies*, 24:270–287, 2012.

[Coelho *et al.*, 2013] Leandro C Coelho, Jean-François Cordeau, and Gilbert Laporte. Thirty years of inventory routing. *Transportation Science*, 48(1):1–19, 2013.

[Cordeau *et al.*, 2015] Jean-François Cordeau, Demetrio Laganà, Roberto Musmanno, and Francesca Vocaturo. A decomposition-based heuristic for the multiple-product inventory-routing problem. *Computers & Operations Research*, 55:153–166, 2015.

[Dantzig *et al.*, 1954] George Dantzig, Ray Fulkerson, and Selmer Johnson. Solution of a large-scale traveling-salesman problem. *Journal of the Operations Research Society of America*, 2(4):393–410, 1954.

[Desaulniers *et al.*, 2015] Guy Desaulniers, Jørgen G Rakke, and Leandro C Coelho. A branch-price-and-cut algorithm for the inventory-routing problem. *Transportation Science*, 50(3):1060–1076, 2015.

[Gurobi Optimization, 2018] LLC. Gurobi Optimization. *Gurobi optimizer reference manual*, 2018.

[Helsgaun, 2017] Keld Helsgaun. An extension of the lin-kernighan-helsgaun TSP solver for constrained traveling salesman and vehicle routing problems. Technical report, Roskilde University, 2017.

[Lai *et al.*, 2018] Xiangjing Lai, Dong Yue, Jin-Kao Hao, and Fred Glover. Solution-based tabu search for the maximum min-sum dispersion problem. *Information Sciences*, 441:79–94, 2018.

[Lin and Kernighan, 1973] Shen Lin and Brian W Kernighan. An effective heuristic algorithm for the traveling-salesman problem. *Operations Research*, 21(2):498–516, 1973.

[Park *et al.*, 2016] Yang-Byung Park, Jun-Su Yoo, and Hae-Soo Park. A genetic algorithm for the vendor-managed inventory routing problem with lost sales. *Expert Systems with Applications*, 53:149–159, 2016.