Hybridization of tabu search with feasible and infeasible local searches for periodic home health care logistics

Ran Liu, Xiaolan Xie, Thierry Garaix

Hybridization of Tabu Search with Feasible and Infeasible Local Searches for Periodic Home Health Care Logistics

Ran LIU [a,b], Xiaolan XIE [a,b], Thierry GARAIX [a]

[a] Center for Health Engineering & LIMOS-ROGI CNRS UMR 6158
Ecole Nationale Supérieure des Mines de Saint Etienne
158 Cours Fauriel, 42023 Saint Etienne, France

[b] Department of Industrial Engineering & Logistics Management
Shanghai Jiao Tong University
800 Dong Chuan Road 200240 Shanghai, China

Corresponding author: Prof. Xiaolan XIE, Ecole Nationale Supérieure des Mines de Saint Etienne, 158 Cours Fauriel, 42023 Saint Etienne, France. Phone: 33-4-77426695, Fax: 33-4-77420249, Email: xie@emse.fr

Abstract: This paper addresses a periodic vehicle routing problem encountered in home health care (HHC) logistics. It extends the classical Periodic Vehicle Routing Problem with Time Windows (PVRPTW) to three types of demands of patients at home. Demands include transportation of drugs/medical devices between the HHC depot and patients' homes, delivery of special drugs from the hospital to patients, and delivery of blood samples from patients to the lab. Each patient requires a certain number of visits within a planning horizon and has a set of possible combinations of visit days. Daily routing should meet time window constraints associated with patients, the hospital and the lab. The problem consists in determining the visit days of each patient and vehicle routes for each day in order to minimize the maximal routing costs among all routes over the horizon. We propose a Tabu Search method combined with different local search schemes including both feasible and infeasible local searches. The proposed approaches are tested on a range of instances derived from existing Vehicle Routing Problem with Time Window (VRPTW) benchmarks and benchmarks on special cases of our problem. Numerical results show that local search scheme starting with an infeasible local search with a small probability followed by a feasible local search with high probability is an interesting hybridization. Experiments with field data from a HHC company show that the proposed approach reduces the total cost and better balances the workloads of vehicles.

Keywords: Home health care, Periodic vehicle routing, Tabu search, Infeasible local search, Feasible Local search

1 Introduction

In this paper, we consider a special periodic vehicle routing problem with time windows constraints arising in the home health care industry in France. Home Health Care (HHC) is a growing medical service in France. The objective of the HHC operation is to provide high quality services to the patients at home in order to help them recover from the illness or injury in a personal environment. This paper addresses the logistic issues in HHC operation. The typical logistic services in the HHC involve delivering the medicines and medical instruments to patients, picking up the biological samples from patients at home and bring them to a medical laboratory, collecting medical waste from patient's home and bring them back to dispose, etc. Such distributing and collecting jobs are served by the HHC company for a large number of patients stayed at their own home. From the view of the HHC company, the core component in the home health care logistics is to find a feasible working schedule to their drivers and vehicles, so as to satisfy the requirements of patients, reduce operating cost, and improve service quality. Apart from these vehicle routing decisions, planning decisions also need to be made to determine the days each patient is served as patients of an HHC often require more than one delivery or pickup services. In practice, HHC usually builds a weekly logistics plan that of course needs to be adapted to face random events such as emergency demands.

Roughly speaking, the HHC logistics problem considered in this paper consists in assigning visit days for each patient to meet demands of patients and designing vehicles routes for each day to visit each assigned patient during that day within a specified time window. Some patient visits are preceded by a hospital visit to pick up special drugs or followed by a lab visit to drop blood samples. Clearly, assigning suitable visiting days to each patient and optimizing these repetitive operations can generate significant cost savings for the HHC logistics operation. In this paper, we address this special optimization problem in HHC logistics and call it the *Periodic Home Health Care Pickup and Delivery Problem* (PHHPDP).

The PHHPDP is similar to the *Periodic Vehicle Routing Problem with Time Windows* (PVRPTW) [1], a combination of the *Vehicle Routing Problem with Time Windows* (VRPTW) and the *Periodic Vehicle Routing Problem* (PVRP). The PVRPTW considers how to serve customers during a planning horizon under time window constraints. Although the VRPTW and PVRP have received considerable attention both in theoretical research and in real world applications, the literature on the PVRPTW is rather limited.

Our PHHPDP has its own characteristics and cannot be solved as a PVRPTW directly due to the following reasons. First, in the PVRP and PVRPTW, vehicles only take goods from the depot to each client to satisfy its demand. In our problem the distribution and collection tasks are more complex. According to the origin and destination of the transportation requirements,

there are four types of logistic demands faced by the HHC company: (1) distribute drugs/medical devices from the HHC depot more precisely the HHC pharmacy to patients' homes; (2) collect the load (unused drugs/medical devices) from the patients' homes back to the depot; (3) deliver special drugs, from a hospital to patients, e.g., chemotherapy drugs for cancer treatment; (4) pick up the blood samples from patients' home to the lab. A patient may have different types of demands, simultaneously. For example, a patient's daily request may consist of both getting the medicines from the hospital, and sending the biological samples from his home to the lab.

In practice, workload balancing among different vehicles/drivers is as important as minimizing the total travel distance usually optimized in the literature. We adopt a special objective in our problem, minimizing the length of the longest route in the planning horizon. Similar objective function is called *min-max* in the VRP and multiple traveling salesman problem (*m*-TSP). The reader is sent to [2] for a survey on *m*-TSP, and to [3] for an application in newspaper printing industry, where a parallel machines with sequence dependent setups problem is modeled as a *m*-TSP with workload balancing.

In most HHC applications the vehicle capacity is hardly a limiting factor as goods under consideration (e.g., a box of medicine, a piece of blood sample) often have small size. For this reason, we assume unlimited vehicle capacity in our PHHPDP. Based on this condition, another important logistics problem *TSP with Time Windows* [4] can be seen as a special case of PHHPDP with one day planning horizon, one vehicle and no hospital and lab visits. Since the TSP with time windows has been proven to be NP-hard, and finding a feasible solution is NP-complete [5], the PHHPDP is also NP-hard.

In this paper, we will build several Tabu Search (TS) algorithms to address this special periodic vehicle scheduling problem. The TS scheme is similar to that of Cordeau et al. [6] with some innovative elements: (1) an augmented criterion taking into account constraint violations with penalty factors dynamically adjusted according to the feasibility of the resulting solution, (2) neighborhood search with both inter-route and intra-route local searches, (3) combination of feasible and infeasible local searches. Especially, numerical results show that infeasible local search with small probability followed by feasible search with high probability is an interesting combination in TS.

The rest of this paper is organized as follows. Section 2 is a survey of relevant literature. Section 3 gives the notation and problem definition. Section 4 proposes TS algorithms for solving the problem. Section 5 presents the computational experiments on the instances derived from existing VRPs benchmarks and on real-life data. Finally, Section 6 presents the conclusions and future research directions.

2 Literature review

Despite the importance of HHC services, only a few papers deal with the HHC problems. Begur et al. [7] designed a decision support system for home health care in United States. Classical savings algorithm and nearest neighbor heuristics were used for optimizing routes. Cheng and Rich [8] studied a HHC model of scheduling full time nurses and part time nurses. The problem, similar to the Multi-depot VRPTW, is to find an optimal schedule such that each nurse leaves from his/her home, visits a number of patients within their time windows, and return home. Two mixed integer programming models and a two-phase construction heuristic were proposed. Bertels and Fahle [9] solved their HHC problem with a hybridization of constraint programming and meta-heuristics including simulated annealing and tabu search. In the decision support system LAPS CARE by Eveborn et al. [10], the HHC problem is formulated as a set partitioning problem with the objective of matching visits to staff members and solved by repeated matching algorithm. Kergosien et al. [11] addressed an assignment and routing problem of HHC workers to care activities. The problem is equivalent to the *m*-TSP with time windows under some specific constraints. An integer linear program was proposed and solved with a commercial solver. Two more recent research are formed by Trautsamwieser et al. [12] and Nickel et al. [13]. Trautsamwieser et al. [12] considered the HHC services problem during natural disasters (especially flood disaster) in Austria. The problem is formulated as a rich VRP with state-dependent breaks in order to minimize the sum of driving times and waiting times, and the dissatisfaction levels of clients and nurses. A mathematical formulation and a variable neighborhood search based approach were proposed for the daily HHC problem. Nickel et al. [13] considered routing and scheduling problems arising in the context of HHC services in Germany. A two stage approach was proposed to determine an optimal weekly service plan. A constraint programming heuristic generates a weekly schedule by minimizing the number of nurse visiting tours. Different heuristic approaches then modify and improve the initial solution to incorporate changes into existing plan. With two real-world data sets they showed the benefit of using the proposed approaches in HHC context.

Compared with existing researches, our study considers HHC operations from a new perspective. We focus on picking up and delivering materials and goods (e.g., medicines, medical instruments, and biological samples) among HHC depot, patient homes, medical laboratory, and hospital. To our best knowledge, our paper is the first to incorporate the schedule of visiting medical laboratory, and hospital in the HHC service problem.

As mentioned above, our PHHPDP model is similar to the PVRPTW. As the PVRPTW has attracted little attention in the literature, we focus our review on the PVRP and its variants. The PVRP has been widely studied in the literature. The first problem motivating the PVRP was introduced by Beltrami and Bodin [14]. The PVRP was formally defined by Russell and

Igo [15] as the ‘assignment routing problem’, and first formulated by Christofides and Beasley [16]. Early heuristics for the PVRP focused on classical construction heuristics [15] [16]. From the mid of 1990s, some meta-heuristics have been proposed. Chao et al. [17] developed a two phase, record-to-record travel algorithm for the PVRP. Cordeau et al. [6] proposed a sophisticated tabu search for the PVRP, which allows infeasible solutions during the search process. Mourgaya and Vanderbeck [18] constructed an approximate solution for the PVRP using a truncated column generation procedure followed by a rounding heuristic. Hemmelmayr et al. [19] and Pirkwieser and Raidl [20, 21] adopted variable neighborhood search methods for the PVRP. Pirkwieser and Raidl [22] presented a column generation approach for obtaining strong lower bounds to the PVRP with time windows. Then, Pirkwieser and Raidl [23] investigated two new variants of heuristics and tested them on the PVRP with time windows, in which variable neighborhood search and evolutionary algorithm were combined with parts of a column generation approach. Gulczynski et al. [24] developed a heuristic for the period vehicle routing problem by using an integer program and the record-to-record travel algorithm. Vidal et al. [25, 26] proposed hybrid genetic algorithms for the PVRP and multi-depot PVRP. Very recently, Cacchiani et al [27] presented a new hybrid optimization algorithm and apply it to solving PVRP and PTSP. This algorithm is based on the linear programming relaxation of a set-covering-like integer linear programming formulation of the problem with additional constraints. A recent sophisticated exact method for the PVRP has been proposed by Baldacci et al. [28].

Besides the basic PVRP, some variants have also been presented and studied. Lacomme et al. [29] introduced and solved a problem called periodic capacitated arc routing problem, where the vehicles must serve a set of arcs in the graph. Cornillier et al. [30] developed a heuristic for the periodic petrol stations replenishment problem in order to maximize the total profit equal to the revenue, minus routing costs and regular and overtime costs. Angelelli et al. [31] and Wen et al. [32] studied the dynamic PVRP in which customer orders are dynamically revealed over time. Angelelli and Grazia Speranza [33] studied an extension of the PVRP where vehicles can renew their capacity at some intermediate facilities. Francis et al. [34] considered a special PVRP in which service frequency is a decision of the model. Gulczynski et al. [24] addressed the PVRP while considering reassigning customers to new routes, and balancing the workload among drivers across routes. When only one vehicle is available every day and vehicle capacity and traveling duration are not considered, the PVRP becomes the *Periodic Traveling Salesman Problem* (PTSP). Some heuristics for the PVRP can be adopted for the PTSP. Specialized heuristics for the PTSP can be found in [6, 16, 19, 24].

Compared with the PVRP, the PVRPTW receive much less attentions. Cordeau et al. [1] introduced this problem and designed a tabu search to solve it. Recently, Yu and Yang [35] used an ant colony optimization to solve the PVRPTW.

In our research the objective is to minimize the length of the longest of all the routes, i.e., the min-max objective. In the field of the VRPs, PVRPs and m-TSPs, the research with the min-max objective is very limited. The m-TSP is a special case of the VRP with unlimited vehicle capacity. França et al. [36] proposed a tabu search algorithm for the min-max m-TSP, which minimizes the cost of the most expensive route among all salesmen. Somhom et al. [37] and Modares et al. [38] developed self-organizing artificial neural network approaches for the m-TSP with min-max objective function. Arkin et al. [39] proved the NP-hardness of the min-max VRP and provided constant ratio approximation algorithms. Golden et al. [40] proposed a tabu search based adaptive memory procedure for both the VRP and m-TSP with min-max objective. Valle et al. [41] investigated an interesting *min-max* selective VRP, where not all customers have to be served.

Although a large number of methods have been proposed for periodic VRPs, e.g., PVRP, PTSP and PVRPTW, we find that all these research try to generate routes that minimize total vehicle traveling distance or time, or the number of the vehicles, etc. To the best of our knowledge, no literature considers the min-max VRPs with time windows in a planning horizon, i.e., min-max PVRPTW. Actually, even neglecting time windows constraints, the remaining min-max PVRP has never been considered in the existing literature.

Note that the PHHPDP can be seen as a special kind of Pickup and Delivery problem (PDP) [42-44]. For example, we can split the hospital (lab) into several demand-based auxiliary nodes; each one represents the original of a patient needing medicines from hospital (who has bio samples to be send to the lab). Then, the PHHPDP can be transformed into a Periodic Pickup and Delivery Problem with Time Windows at the cost of an artificial increase of the problem size. Although some heuristics [45] and exact approaches [46, 47] have been designed for the PDP with Time Windows, there is no research about the Periodic PDPTW. Even for a relatively simple version, the Periodic PDP, we cannot find any literature about this problem.

3 Notation and problem definition

The PHHPDP in home health care logistics is defined formally on a graph as follows. Let $G = (V, A)$ be a graph, with node set $V = \{0, 1, ..., n, n+1\} \cup \{h, l\}$ and arc set $A=\{(i, j): i, j \in V, i \neq j\}$. $N = \{1, ..., n\}$ denotes the set of patient locations, h and l denote a hospital and a lab, nodes 0 and $n+1$ the origin and destination depots of the home health care company. Node $n+1$ is the same location as node 0, implying that each vehicle starts and ends at the depot. The location of each node (its x- and y-coordinates) is known. A homogeneous fleet of K vehicles, initially located at the depot, is available to serve the patients. Vehicle capacity is not considered, as it is hardly the limiting constraint in practice.

There are three classes of patients. A patient of class 1 denoted as P_1 requires delivery

from the depot to home or pick up from home to the depot. A patient of class 2 denoted as P_2 requires delivery from the hospital to home. A patient of class 3 denoted as P_3 requires pick up from home to the lab. Each service requested by a patient is called a *demand*. A patient may require different classes of demands. For example, for a patient $i \in P_2 \cap P_3$, the HHC company has to pick up blood samples from patient i and bring it to the lab, and deliver drugs from the hospital to this patient.

Each node $i \in N$ is associated with a time window $[a_i, b_i]$, where a_i, b_i represent the earliest and latest service time. The depot node also has a time window, representing the earliest departure time and the latest return time. Each arc $(i, j) \in A$ has a routing cost c_{ij} and a traveling time t_{ij}. Without loss of generality, the service time for each node i is included in the traveling time t_{ij} and is not explicitly considered.

A vehicle is allowed to arrive at a location i before a_i and wait until the patient becomes available, but arrivals after b_i are prohibited. Each vehicle starts at time 0 from node 0, travels to the location of the first node i_1 on its route, waits till the availability of the node at a_{i1}, then travels to the location of the second node and so on and so forth till visiting all nodes on its route and returning to the node $n+1$. We call the length of a route the total routing costs of arcs visited by the vehicle. A route is said infeasible if the vehicle arrives at a node i after b_i, or a P_2-patient visit is not preceded by a hospital h visit, or a P_3-patient visit is not followed by a lab l visit. The set of days in the planning horizon is denoted by $D = \{1, \ldots, d\}$, and d represents the number of days. Each patient $i \in N$ has a visit frequency f_i, and a set of allowable service patterns R_i. Each $r \in R_i$ is a subset (combination of days) of the planning horizon D. Patterns of each patient contain the same number of days in which the patient is visited.

The visit frequency, the service patterns, and time windows are defined for patients instead of demands. This implies that all demands of a patient share the same visit frequency and visiting a patient on a given day implies serving all its demands. Extension to patients having demands of different visit frequency and different time windows is addressed at the end of this section.

The PHHPDP consists in selecting a service pattern for each patient and designing daily vehicle routes, such that: (1) visits to patients match to selected patterns, (2) each patient is visited at most once a day, (3) at most K routes starting from and ending at the depot are used each day, (4) each route must satisfy time windows and precedence constraints on each patient node, (5) minimizing the length of the longest route in the planning horizon. As the vehicle capacity is infinite, each vehicle requires at most one daily visit to the lab and to the hospital. Table 1 summarizes the defining notations of the PHHPDP.

Table 1. Defining elements of PHHPDP

D	Planning horizon

N	Patient set
h	Hospital
l	Lab
P_1	Set of patients requiring delivery between the depot and home
P_2	Set of patients requiring drugs from the hospital
P_3	Set of patients having blood samples to be delivered to the lab
a_i, b_i	Earliest and latest visit time of node i
f_i	Visit frequency of patient i
R_i	Set of allowable service patterns of patient i
K	Set of available vehicles
c_{ij}	Routing cost from node i to node j
t_{ij}	Traveling time from node i to node j

The definition of the PHHPDP can be extended to include other operating constraints. We consider patients with demands of different visit frequencies, visit patterns and demand specific time windows. Multiple visits to a patient are also allowed. This extended PHHPDP can be transformed into a basic PHHPDP by transforming a patient into several demand-based fictive patients. For example, for a patient i has two demands p_{i1} and p_{i2} and should be served two and four times weekly respectively, we delete patient i and generate two fictive patients i' and i'', each representing a demand and associated with the related visit frequency and time window. The locations of i' and i'' are the same as that of patient i, i.e., the distance between i' and i'' *is* zero and the distance between i' (i'') and another node j equals the distance of i and j. All time window and precedence constraints can be transformed accordingly..

4 Solution procedure for the PHHPDP

In this section, we propose a tabu search (TS)-based algorithm to solve the PHHPDP. The proposed TS algorithm in Figure 1 is based on the general TS framework developed by Cordeau et al. [6]. Similar attribute set and augmented criterion function for constraint violations have been successfully adapted in TS algorithms to solve some variants of the VRP [48-50]. The algorithm starts from an initial solution s that can be feasible or infeasible. The tabu list and aspiration value of each attribute are then initialized. Neighbor search is applied to solution s by executing some inter-route local moves. The best solution s' is selected from neighbor solutions that are either not tabu or satisfy some aspiration criterion. Solution s' is further modified and improved by intra-route local search methods (Section 4.4). We then update the tabu list, aspiration level of each attribute and some algorithm parameters. The TS restarts from $s=s'$ till a stopping criterion is met.

Although the basic structure of our approach is similar to that proposed in Cordeau et al. [6], there are some key differences with respect to: (1) the construction method of the initial solution, (2) the evaluation of the objective function, (3) the update rules of penalty

parameters, (4) the construction of the neighborhood of a solution taking into account the lab, hospital and precedence constraints. Most important, Cordeau et al. used GENI heuristic to perform insertion and removal of customers from routes to construct the neighborhood. At each iteration, GENI heuristic performs intra-route local search to some routes within the best neighbor solution. In our approach, we adopt standard insertion and removal of nodes to identify inter-route neighborhood search. Furthermore, we apply two intra-route local search strategies to improve and diversify each route in the current solution. In the following, we give the detailed functions of our approach.

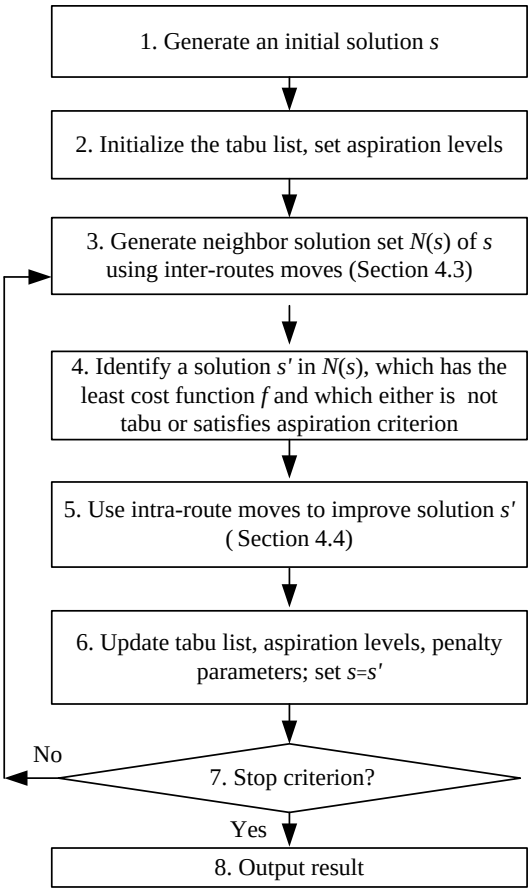

Figure 1. General structure of TS algorithm

4.1 Augmented criterion function

In our approach, each solution corresponds to a set of routes for each day. For each route, the visit times at different nodes are determined by taking into account the visiting sequence and the earliest available time of each node. As both feasible and infeasible solutions are allowed, a solution s is evaluated by an augmented cost function

$$f(s)=c(s)+\alpha\, q(s)+\beta\, g(s)$$

where $c(s)$ is the original objective function, i.e. the length of the longest route, $q(s)$ and $g(s)$ denote the total violation of time window and precedence constraints of all the routes,

respectively, α and β are penalization parameters. $q(s)$ and $g(s)$ are defined as follows:

$$q(s) = \sum_{k \in K} \sum_{i \in U_k} [t_i^k - b_i]^+$$

$$g(s) = N_{hk} + N_{lk}$$

where $[x]^+ = max\{0, x\}$, U_k is the set of nodes visited by vehicle k, t_i^k is the visit time of node i in route k, b_i is the latest allowed visit time of i, N_{hk} (N_{lk}) is the number of P_2 (P_3) patients visited before the hospital visit (after the lab visit) in route k. Clearly, if s is a feasible solution, $q(s)=0$, $g(s)=0$, and $f(s) = c(s)$.

In TS, parameters α and β are dynamically adjusted to facilitate the exploration of the search space. TS algorithm starts from α_0 and β_0, which are set to be 1 and 100, respectively. Meanwhile, we set two intervals $[\alpha_{min}, \alpha_{max}]$ and $[\beta_{min}, \beta_{max}]$ for these two parameters, which limit their maximum and minimum values during the search process. The values of α and β are increased or decreased throughout the iterations. At the end of iteration, if the resulting solution is feasible, α is divided by a factor $1+\varphi_1$. If the resulting solution is infeasible and the time window constraints are violated, α is multiplied by a factor $1+\varphi_1$. If the resulting solution is infeasible but the time window constraints are satisfied, α is divided by a factor $1+\varphi_2$ ($0 < \varphi_2 \le \varphi_1$). Parameter β is adjusted by the same rules. The following values, $\alpha_{min} = \beta_{min} = 0.01$, $\alpha_{max} = \beta_{max} = 1000$, $\varphi_1 = 0.2$ and $\varphi_2 = 0.05$ are used in our approaches.

4.2 Initial solution

We first apply the following heuristic procedure to generate an initial solution of PHHPDP for the TS algorithm. Time windows are not considered here and hence the initial solution can be infeasible.

Step 1: Select randomly a service pattern for each patient.
Step 2: Repeat 3-6 to build the vehicle routes for each day $d' = 1, \ldots, d$.
Step 3: Sort patients of day d' in ascending order of their angular coordinate in the polar coordinate representation with the depot at the origin.
Step 4: Determine the patient i which is closest to the depot. Generate the first route on day d' to serve patient i. Initialize the counter $k=1$ of the route number.
Step 5: Repeat 6 for each patient j in cyclic order of Step 2 starting from patient i.
Step 6: Insert j into a route $k' \in \{1, \ldots, \min(K, k+1)\}$ with the minimal route length after insertion of j.

At Step 1 each patient is assigned a random pattern. We then solve the VRPs with min-max objective on each day by neglecting time windows. We first generate in Step 4 a route from the depot to the closest patient i. If this patient i belongs to P_2 or P_3 set, the first

route is (depot-*h*-*i*-depot) or (depot-*i*-*l*-depot) in order to include the corresponding hospital or lab visit. The other patients are then inserted in step 6 in ascending order of their angular coordinate by starting from patient *i*. Each patient *j* is inserted in an existing route or a new one such that the length after its insertion of the selected route is minimized. The position in a route of the patient *j* to insert is determined by exhaustive search. For the insertion of a P_2 or P_3 patient in a route *k*, the position of the hospital or lab must be considered simultaneously by exhaustive search. If *j* is the first P_2 patient in route *k*, we should also insert the hospital visit at the least cost position. If a hospital already exists in the route *k*, we also try to relocate it at each possible position (before all P_2 patient) when inserting patient *j*.

4.3 Attribute set, tabu list, aspiration and stopping criterion

Generally, TS utilizes some form of adaptive memory, called tabu list and tabu duration, to implement a diversification strategy. In our approach, each solution is characterized by an attribute set $B(s)=\{(i, k, d)|_{i \in N, k \in K, d \in D}$: patient *i* served by vehicle *k* on day *d*$\}$. A neighbor of a solution *s* is obtained by applying an operator that deletes a set of attributes from $B(s)$, and replaces it by a new set of attributes. Then, when a patient *i* is removed from a route *k* on day *d*, we assign a tabu status to this attribute (i, k, d), and set a tabu duration θ to this attribute. In the next θ iterations, re-inserting patient *i* back into route *k* on day *d* is forbidden. The tabu duration θ takes the values in $[\theta_{min}, \theta_{max}]$ and starts from θ_0. It dynamically modifies during the search process: (1) after each improvement of the current best solution, θ is set equal to θ_{min}; (2) after $\theta_{\emptyset}$ consecutive unimproved iterations, θ is updated to be $min(\theta+1, \theta_{max})$.

One simple aspiration criterion is adopted in TS. Each attribute is associated with an aspiration value, which is defined as the cost of the best feasible solution found with that attribute. Thus, a neighbor solution $\bar{s}$ of the current solution *s* can be considered only when: (1) all new attributes which are not in *s* but in $\bar{s}$, are non-tabu, or (2) $\bar{s}$ is feasible and $f(\bar{s})$ is less than the aspiration values of these attributes.

Three stopping criterion in TS is: (1) after a fixed number N_1 of iterations, or (2) after a fixed number N_2 of iterations without improving the current best solution, (3) after a fixed total running time *T*.

4.4 Neighborhood

Essentially, TS algorithm keeps on finding the best neighbor of the current solution. A solution that can be obtained from a given solution using an allowable move is called a neighbor. Three inter-route local moves are considered in our TS algorithm:

(1) Remove a customer *i* from vehicle route *k*, and insert it at least cost into another route k' on the same day, $k' \neq k$. Note here route k' may be nonempty or empty route. All customers *i* and destination route *k'* are considered each day. The size of this move is $O(n \cdot d)$.

(2) Exchange a patient i in vehicle route k and another patient j in vehicle route k' on the same day with $k' \neq k$. All pairs of customers (i, j) are considered each day, i.e. $O(n^2 \cdot d)$ moves.

(3) Replace the visit pattern $r \in R_i$ currently assigned to customer i with another pattern $r' \in R_i$. For each day d' in D, if day d' belongs to pattern r but not r', customer i is removed from its route in day d'. If all the customers in this route are removed due to this operator, the route is deleted from the solution. Meanwhile, if d' belongs to pattern r' but not r, customer i is inserted into route k' in day d' while the increase in $f(s)$ is minimized. Here, we allow route k' to be a nonempty or empty route. The size of this move is $O(\sum_{i=1}^{n} |R_i| \cdot |r|_{r \in R_i} \cdot n)$

The neighborhood $N(s)$ of a solution s consists of all the solutions that can be obtained by performing one of the foregoing transformations. We should point out that, during each local move when a patient is deleted from or inserted into a route, we do not modify hospital and lab visits in this route. For example, when inserting a patient in route k', we do not remove, relocate or insert the hospital visiting or the lab visiting in the route even if it is necessary to satisfy the new patient. When the best neighbor is obtained, we deal with each modified route in this neighbor as follows: (1) we remove the un-necessary hospital or lab visit from a modified route. That is to say, if there is no more P_2 (P_3) patient in route k after the move, we remove the hospital (lab) in route k, (2) we insert the missing and necessary hospital or lab visit into the modified route. For example, when a patient i is inserted into route k' and i is the only P_2 (P_3) patient in this route, we should add the hospital (lab) into route k'. We insert the lab and hospital into the route at feasible positions with the smallest increment augmented cost. At this step, the precedence constraints are satisfied and $g(s)=0$, i.e. the hospital is visited before all P_2 patients and the lab is visited after all P_3 patients.

4.5 Hybrid Tabu Search with Local Search

Our approach hybridizes a Tabu Search algorithm and local search procedures. The Tabu Search procedure applies inter-routes movements between pair of routes, as shown in Section 4.4. Then, the solution obtained by Tabu Search inter-routes movements is further improved by means of intra-route Local Search procedures. Such combination of local search procedures has been proved to be an effective strategy to improve the performances of meta-heuristics. For example, Yu et al. [51] designed an improved ant colony system algorithm to solve the VRP, in which intra-route search is adopted to improve individual routes of current solution during the iteration. Jozefowiez et al. [52] proposed an evolutionary algorithm for the VRP with route balancing, and an intra-route local search (2-opt) was chosen to improve each route of each offspring solution. Some researchers adopted both intra-route and inter-route search in their algorithms for the VRP [53-55]. Similarly, in a weighted tardiness minimization problem of parallel machines, Della Croce et al. [56] applied local search on each machine independently at each algorithm iteration. Concerning the Tabu

Search, the GENI heuristic in [6] plays a role as intra-route local search. Similar way was used in the tabu search [57] for solving an open VRP. In Brandão's tabu search, each iteration of the TS modifies only two routes of the current solution. Two simple heuristics, the Nearest Neighbor method and the Unstringing and Stringing procedure, were applied in Gendreau et al. [58] to improve two modified routes. Both our approach and the TS of [6] use inter-route local search to get the best solution s' from $N(s)$. However, we adopt two special and new local search strategies to improve each route independently, not only for the modified routes. The first type of local search is called *Infeasible Local Search* (ILS) and the second *Feasible Local Search* (FLS).

FLS is the widely used classical local search method. It can be used to improve route construction methods [59], or be hybridized with meta-heuristics [60, 61]. FLS starts from a feasible solution and improves it by local moves. Once a neighbor solution is identified, it is compared against the current solution. If the neighboring solution is better, it replaces the current solution, and the search procedure continues. In the FLS, each neighbor solution must be feasible. Compared with FLS, ILS can be applied to both feasible and infeasible solutions. Meanwhile, during the ILS search, both feasible and infeasible neighbor solutions may be generated.

In the paper, the FLS starts from a feasible seed solution and improves each route by using 1-1exchange, 1-0 relocation, 2-Opt exchange. The 1-1 exchange tries to exchange the positions of two nodes (patient, lab and hospital) in a route. The 1-0 exchange is the relocation of one node, i.e., transferring a node from its position to another position in the same route. The 2-opt exchange tries to improve the route by replacing two of its edges (i, i+1), (j, j+1) by two new edges (i, j) and (i+1, j+1). The first-accept strategy is used, i.e. once a feasible and better route is found, it is adopted as the new seed for repeating the local search. The whole local search stops when no additional improvement can be obtained. Note that during the search procedure, time window and precedence constraints must be satisfied. For each route o, FLS uses its real distance $c(o)$ to evaluate the cost of a local move. Compared with FLS, infeasible moves are allowed in the ILS, i.e., the time window and precedence constraints can be violated at each move. For each route r, ILS uses the augmented criterion function $c(o)+\alpha q(o)+\beta g(o)$ to evaluate the cost of a local move, where $q(o)$ and $g(o)$ denote the violation of time window and precedence constraints of this route, respectively. Note that even ILS starts from a feasible seed solution it may generate an infeasible result when the ILS procedure completes.

FLS and ILS play different roles in our method. FLS is used to intensify TS algorithm, just like the classical local search procedure integrated in other meta-heuristics. ILS can be seen as a way for diversifying the search of TS method. For example, when the algorithm sinks into a local optimal solution, ILS may generate a new infeasible neighbor solution and

leads to a new search direction. For these reasons, we give two probabilities, p_{FLS} and p_{ILS}, for applying FLS and ILS, respectively. Meanwhile, to test different strategies of applying the FLS and ILS, we design and test following five TS strategies:

TS_C	TS without any additional local search, i.e., $p_{FLS} = p_{ILS}=0$.
TS_{FLS}	Improve feasible current solution s' with FLS, i.e., $p_{FLS}=1$ and $p_{ILS}=0$.
TS_{ILS}	Improve current solution s' with ILS, i.e., $p_{FLS}=0$ and $p_{ILS}=1$.
$TS_{F\text{-}I\text{-}P}$	Use either FLS or ILS to the current solution s' according to its feasibility. If s' is feasible, FLS is used; otherwise, ILS is applied.
$TS_{I\text{-}F\text{-}S}$	Use FLS and ILS sequentially with probabilities p_{ILS} and p_{FLS}. It first uses ILS with probability p_{ILS}. If the resulting s' is feasible, FLS is used with probability p_{FLS}.

To our best knowledge, such hybridization scheme has never been proposed for solving the VRP and relative problems. Thus, in our experiments, we will intensively test and compare these five tabu search algorithms.

5 Computational experiments

This section presents computational experiments designed to assess the performance of the proposed method. Since there is no benchmark data available for the problem of this paper, we construct some test instances based on existing VRPTW benchmarks. We also test our algorithm on the classical *min-max* Multiple Traveling Salesman Problem (*min-max* MTSP), which can be seen as a special case of our problem with a planning horizon of one day and without hospital, lab and time windows. We extend TS algorithm and test it on another rather classical problem, Periodic Traveling Salesman Problem (PTSP), which is similar to our problem but with only one vehicle, with a different objective function and without hospital and lab visit. Concerning the latter two problems, our approaches are compared with the state of the art algorithms. Finally, we also compare solutions obtained by our algorithm against real-life routing plan built by a French home health care company.

All algorithms of this paper are implemented in C on a 3.2 GHz Dual Core computer with a 2 GB memory under Linux. All tabu search algorithms run 10 times for each test instance. The best, the average, the worst results and the average running time are obtained from 10 runs for each TS algorithm, and used to assess the performances of these algorithms. Table 2 summarizes the parameter setting of the algorithms used in the computational experiments.

Table 2. Parameter setting in the experiment

Symbol	*Explanation*	*Value in experiment*
α_0, α_{min}, α_{max}	Initial, minimum and maximum values of α	1, 0.01, 1000
β_0, β_{min}, β_{max}	Initial, minimum and maximum values of β	100, 0.01, 1000
φ_1, φ_2	Parameters for updating α , β	0.2, 0.05
θ_ϕ	Number of consecutive unimproved iterations to update tabu duration	30
θ_0	Initial tabu duration	$\max\left(\lfloor 4\log_{10} n \rfloor, 7\right)$
$\theta_{min}, \theta_{max}$	Maximum and minimum values of tabu duration	$\theta_0 - 4$， $\theta_0 + 8$.

5.1 PHHPDP instances derived from VRPTW benchmarks

We first derive test instances from existing VRPTW benchmarks of Solomon [62], and Gehring and Homberger [63]. Solomon's VRPTW instances are divided into three classes that differ by the geographical distribution of the customers: *C*, *R* and *RC* type instances. Each class is divided into two series: the 100-series instances with tighter time windows and the 200-series instances with wider time windows. We select 6 *C* type instances, 6 *R* type instances and 6 *RC* type instances. Among 6 instances of each type, both the 100-series instances and 200-series ones exist. Based on each selected Solomon instance, we derive 3 new instances for our problem with 3Z patients as follows. We randomly choose 3Z customers from the Solomon instance as the P_1 patients, then, Z P_2 and Z P_3 patients are randomly selected from P_1 patients. Therefore, the generated instances contain 5Z demands required by 3Z patients. The depot is located as in Solomon instances at (40, 50) for C-type and RC-type instances and at (35, 35) R-type instance, and the locations of lab and hospital are (90, 50) and (10, 15). For each Solomon-based instance, the planning horizon is 7 days, and service frequency is generated uniformly in [1, 5]. The service days are randomly selected in the 7 days. For each patient, the time window in Solomon instance is used directly. Concerning the depot, lab and hospital, the earliest times of their time windows are inherited from Solomon's depot time window. The latest time of depot time window in the Solomon instance is multiplied by 120%, and assigned to the depot, lab and hospital in our test instance as the end of new time window. Even so, there may exist some 'violative' P_3 (or P_2) patients, e.g., even if a vehicle starts from depot and goes to a P_3 patient directly, then goes to lab and returns back to the depot, this vehicle still breaks the time window of the depot (later than the end of the depot time window). If such violative patients exist in our instance, the latest time of time windows of lab, hospital and depot are multiplied by 120% again until all violative patients are eliminated. In the preliminary experiment, we find that violative patients are eliminated after two tries.

For each Solomon instance, the above constructing procedure is repeated twice, generating one small size (Z=10 and total 50 demands), one medium size (Z=20 and total 100 demands) instance. For each small/medium size instance, 10 and 15 vehicles are available,

respectively. For these small and moderate test instances, the stopping criteria of tabu search algorithms are: N_1=15000, N_2=9000 and T=∞, i.e., the whole search stops after 15000 iterations, or 9000 unimproved iterations.

Besides Solomon's instances, we also create 18 large instances from VRPTW instances of Gehring and Homberger [63]. These VRPTW instances are similar to but larger than Solomon instances, containing hundreds of customers. We choose 6 instances from Gehring and Homberger VRPTW benchmarks, each of which contains 400 customers. Each instance undergoes the same procedure described above three times with Z=70. The coordinates of the depot are kept at (100, 100), and the locations of lab and hospital are still (90, 50) and (10, 15). Thus, we generate 18 new instances for our study. Each of these large instances contains 210 patients and 350 demands. For each instance, 15 vehicles are available, the planning horizon is 7 days, and the maximal service frequency for each patient is 3 times. For these large size instances, in order to save the computational time, we reduce the maximum number of iterations in the TS, i.e., N_1=5000, N_2=3000 and T=∞.

5.2 Probabilities of feasible and infeasible local searches, and penalty parameters update scheme

We first conduct some experiments to find appropriate probability parameters, p_{FLS} and p_{ILS}, in $TS_{I\text{-}F\text{-}S}$ approach. In a preliminary experiment, we find that the performance of TS can be improved by using a relative high probability of FLS improvement methods. Thus, we set p_{FLS} equal to 80%. To the best of our knowledge, there is no research about the probability of applying infeasible local search procedure. Among the instances generated from Solomon VRPTW benchmarks, we select 10 instances randomly and apply $TS_{I\text{-}F\text{-}S}$ to these instances with different values of p_{ILS}: 0%, 2%, 5%, 10%, 20%, 30%, 40%, and 50%. In order to determine the appropriate p_{ILS} in $TS_{I\text{-}F\text{-}S}$, we adopt four criteria. The first is the number of '*best run*', which represents the number of times a setting ($TS_{I\text{-}F\text{-}S}$ with a special value of p_{ILS}) is able to find the best solution among all the settings ($TS_{I\text{-}F\text{-}S}$ with various values of p_{ILS}). For example, as shown in Table 12, applying $TS_{I\text{-}F\text{-}S}$ with p_{ILS}=2% to instance R210_60 ten times gets the best solution with the cost of 208.84 six times. The other three criteria are the best, the average, and the worst solution costs obtained from 10 runs for each instance. In general, the first criterion is rather stricter than others, particularly when the solution costs obtained by different p_{ILS} settings are similar. Detailed computational results obtained on these 10 instances are presented in Table 12 in the Appendix.

In Table 3, we summarized computational results obtained with eight settings of p_{ILS}. Column p_{ILS} is the value of p_{ILS}, column *BR* the total number of best runs over 100 runs with 10 for each instances, the other columns give the mean values over 10 instances of Best, Average and Worst solutions. We observe the superiority of $TS_{I\text{-}F\text{-}S}$ with small p_{ILS}, e.g., p_{ILS}

equals 2% or 5%, over $TS_{I\text{-}F\text{-}S}$ with high values of p_{ILS}, especially concerning the number of the best run. Therefore, we use p_{ILS}=0.02 and p_{FLS}=0.8 as the stand setting of $TS_{I\text{-}F\text{-}S}$, which are used in every run of $TS_{I\text{-}F\text{-}S}$ on every test instance reported in following sections.

Based on this setting, we also test the way of updating the penalty parameters in the tabu search. In Cordeau et al. [6], different penalty parameters, e.g., parameters for violation of vehicle load and time window constraints, are adjusted with respect to solution's feasibility, simultaneously. That is to say, once a constraint is not satisfied in current solution, all penalty parameters are modified by a factor larger than 1. Otherwise, parameters are divided by this factor. In our approach, each penalty parameter is adjusted according to whether its corresponding constraint is violated or not (See section 4.1). To compare these two updating mechanism, we run $TS_{I\text{-}F\text{-}S}$ with both mechanisms to solve the 10 instances selected above. We compare the results of two updating schemes in Table 4, with two major columns of '*Synchronous update*' and '*Asynchronous update*'. We report the best, average and the worst solution costs of 10 runs for each test instance. Meanwhile, we provide some detailed information about the best solution, i.e., the number of routes, the number of visits to the hospital, the number of visits to the lab, and the number of routes which visit neither lab nor hospital. These numbers are listed in the bracket, beside the best solution cost and separated by oblique line. Our 'asynchronous update' can find 9 better solutions, while 'synchronous update' finds 8 better solutions. Concerning the sum of the best solution costs, the gap is only −0.18%. Such gaps are rather small. But if we focus on '*BR*' column, i.e., the number of best solutions, 'asynchronous update' can find best solution 40 times, while 'synchronous' can only find 29 times. Note that similar 'asynchronous update' is also used in Vidal et al. [25] to dynamically adjust the penalty parameters during the iterations of their (GA) algorithm to favor the generation of naturally feasible individuals.

Table 3. Comparison between different p_{ILS} settings in $TS_{I\text{-}F\text{-}S}$

p_{ILS}	*BR*	*Best*	*Average*	*Worst*
0%	14	211.70	233.86	279.34
2%	40	210.44	214.26	224.87
5%	35	210.74	214.16	218.26
10%	33	211.21	215.06	225.21
20%	28	210.46	216.16	230.22
30%	24	210.39	216.50	231.14
40%	23	211.13	218.63	233.63
50%	23	211.14	219.68	235.48

Table 4. Comparison between different penalty parameter updating mechanisms

	Asynchronous update				*Synchronous update*			
Instance	*Best*	*Average*	*Worst*	*BR*	*Best*	*Average*	*Worst*	*BR*
C101_30	**207.06** (56/26/28 /11)	210.13	233.24	0	207.14 (48/23/23/11)	213.14	233.24	0
C104_30	**175.62** (45/24/16/5)	194.68	221.71	2	**175.62** (56/24/17/15)	195.23	221.62	2

C109_60	**223.39** (48/25/25/9)	223.39	223.39	10	**223.39** (54/34/27/8)	223.39	223.39	10
C201_60	215.20 (56/31/37/6)	219.08	223.70	0	**214.00** (56/36/37/1)	217.25	220.33	0
R108_30	**188.49** (49/27/24/9)	190.26	206.14	9	**188.49** (53/24/24/11)	193.92	229.31	3
R207_30	**234.83** (56/20/22/25)	236.09	245.23	0	**234.83** (54/22/21/22)	236.62	244.02	0
R210_60	**208.84** (56/32/29/11)	211.56	215.63	6	**208.84** (53/34/25/10)	210.97	215.63	5
RC105_60	**193.09** (70/39/39/11)	194.50	197.85	4	**193.09** (70/42/43/11)	216.67	274.42	1
RC201_60	**229.62** (56/38/32/3)	233.26	239.21	0	235.01 (56/35/36/4)	240.37	244.53	0
RC204_60	**228.22** (56/40/25/6)	229.66	242.62	9	**228.22** (55/35/26/10)	230.21	248.07	8
Total	2104.36			**40**	2108.63			**29**

5.3 Computational results on PHHPDP instances

The computational results obtained on the VRPTW-based PHHPDP instances are summarized in Tables 5 and 6. Detailed computational results are presented in Tables 13-15 in the Appendix. Table 5 shows the average min-max objective costs by grouping instances according to the number of demands (column *De*) and the type of the instance (column *Type*). The results are obtained from 10 independent runs for each instance of five approaches, TS_C, $TS_{I\text{-}F\text{-}S}$, TS_{FLS}, TS_{ILS}, and $TS_{F\text{-}I\text{-}P}$. Columns '*Best*', '*Average*' and '*Worst*' present the best, the average and the worst solution costs over 10 runs. Column '*CPU*' gives the average computational time for one run in seconds. Table 6 shows the number of *best run* of each algorithm on each type and scale instances.

We can assert some conclusions from the data in Tables 5 and 6. Firstly, concerning the solution quality, the performance of $TS_{I\text{-}F\text{-}S}$ is better than the other tabu search approaches for different scales and types of test instances. For 48 out of 54 test instances, $TS_{I\text{-}F\text{-}S}$ can find the best solutions among five approaches. Concerning the best solution cost, $TS_{I\text{-}F\text{-}S}$ is better than TS_C, TS_{FLS}, TS_{ILS}, $TS_{F\text{-}I\text{-}P}$ with deviations of 0.33%, 0.27%, 5.12% and 0.50%. The superiority of $TS_{I\text{-}F\text{-}S}$ is even higher with respect to average and worst solution costs. For example, for all test instances the average solution costs of TS_{FLS} and other four approaches deviate by 6.66%, 2.65%, 10.74%, and 1.60%. Thus, we can draw a preliminary conclusion that the solution quality of $TS_{I\text{-}F\text{-}S}$ is the best, followed by TS_{FLS} and $TS_{F\text{-}I\text{-}P}$. The solution quality of TS_{ILS} is not as good as other approaches. The superiority of $TS_{I\text{-}F\text{-}S}$ can also be verified in Table 6. Among 540 runs on all test instances, $TS_{I\text{-}F\text{-}S}$ is able to find the best solution 313 times with respect to 166, 210, 82 and 224 times for TS_C, TS_{FLS}, TS_{ILS}, $TS_{F\text{-}I\text{-}P}$.

Figures 2 and 3 further compare different TS approaches. Figure 2 illustrates the number of '*best run*' of each TS approach over different scale-type instances. As shown in Figure 2, the line of $TS_{I\text{-}F\text{-}S}$ is always the highest one, i.e. $TS_{I\text{-}F\text{-}S}$ has the highest number of '*best run*' for different scale-type of instances. Figure 3 illustrate the mean value of the average solution costs among 10 independent runs for different instances of the same scale and the same type, for different scale-type instances. The line of $TS_{I\text{-}F\text{-}S}$ is always the lowest one, i.e. $TS_{I\text{-}F\text{-}S}$ has the ability of finding the smallest cost of solutions.

Apart from the $TS_{I\text{-}F\text{-}S}$ approach, TS_{FLS} and $TS_{F\text{-}I\text{-}P}$ also generate good results. We do not recommend TS_C since its worst solution cost over 10 runs for each instance is significantly worse than best approaches. As shown in Table 5, for all 54 test instance, concerning the worst solutions costs, TS_C is worse than TS_{FLS} and $TS_{F\text{-}I\text{-}P}$ with deviations of 11.33% and 15.20%. Comparing TS_{FLS} with $TS_{F\text{-}I\text{-}P}$, there is no clear enough computational evidence for choosing one strategy instead of the other. Over 54 test instances, the best solution cost of TS_{FLS} is slightly better than that of $TS_{F\text{-}I\text{-}P}$ with a deviation of 0.22% on the average best solution cost. However the average and worst solution costs of $TS_{F\text{-}I\text{-}P}$ are better than those of TS_{FLS} with mean deviations of 1.07% and 4.35%.

Note that the improved solution quality of $TS_{I\text{-}F\text{-}S}$ is obtained at the cost of longer computation time. Over total 54 test instances, the CPU time of $TS_{I\text{-}F\text{-}S}$ is on average 17.02%, 13.90%, 13.51% longer than that of TS_C, TS_{FLS} and $TS_{F\text{-}I\text{-}P}$, respectively. The differences of computation time between $TS_{I\text{-}F\text{-}S}$ and other approaches are especially true for problem instances of the largest size. We also try to extend the running time of TS_C, TS_{FLS} and $TS_{F\text{-}I\text{-}P}$ to that with $TS_{I\text{-}F\text{-}S}$ for some randomly selected sample instances. The superiority of $TS_{I\text{-}F\text{-}S}$ remains true and TS_{FLS} and $TS_{F\text{-}I\text{-}P}$ are still the next two best approaches.

These experimental results also indicate that the local search procedures play a useful role in the TS algorithm. The performance of basic TS (TS_C) can be improved by integrating the classical local search FLS (TS_{FLS}). Further improvement can be obtained by combining FLS and ILS whether sequentially ($TS_{I\text{-}F\text{-}S}$) or in parallel ($TS_{I\text{-}F\text{-}P}$). Of course, we must point out that only using IFS in basic TS (TS_{ILS}) leads to the deterioration of TS performance.

Table 5. Average routing costs of 10 independent TS runs on PHHPDP instances

		TS_C				$TS_{I\text{-}F\text{-}S}$				TS_{FLS}				TS_{ILS}				$TS_{F\text{-}I\text{-}P}$			
De	*Type*	*Best*	*AVG*	*Worst*	*CPU*	*Best*	*AVG*	*Worst*	*CPU*	*Best*	*AVG*	*Worst*	*CPU*	*Best*	*AVG*	*Worst*	*CPU*	*Best*	*AVG*	*Worst*	*CPU*
50	*C*	208.89	233.09	297.12	52.4	**207.46**	211.36	218.93	55.8	211.40	232.04	313.13	58.0	208.11	216.66	238.67	54.2	207.69	214.04	230.15	53.2
	R	192.91	208.31	231.92	43.6	192.82	198.15	208.78	50.2	192.82	202.99	213.74	48.2	192.82	197.98	207.06	52.6	**192.80**	198.53	208.76	50.7
	RC	206.90	223.04	261.79	55.4	**203.49**	207.36	214.68	61.4	209.10	215.86	231.87	50.2	203.70	211.02	218.24	62.1	210.00	212.42	220.56	55.7
100	*C*	223.65	245.16	299.73	227.5	**220.05**	221.50	222.75	296.2	220.34	233.61	249.94	239.5	234.98	260.10	290.46	273.2	221.84	228.39	245.46	224.7
	R	205.74	212.02	228.27	251.6	**205.74**	206.29	207.79	293.7	205.80	215.22	239.09	250.3	207.34	215.88	234.15	301.0	**205.74**	208.76	217.39	276.9
	RC	226.61	242.48	265.34	263.2	227.21	228.51	230.62	329.1	**225.88**	236.11	253.83	277.6	235.54	260.24	298.01	288.3	231.68	244.50	264.21	284.8
350	*C*	514.64	557.13	684.48	5299.4	515.08	520.36	531.72	6014.5	515.08	524.48	552.91	5354.9	546.66	608.39	679.61	5292.1	**514.13**	522.78	539.53	5083.2
	R	556.41	600.44	677.15	4631.9	554.42	562.50	578.01	5831.5	**553.73**	569.55	614.83	4605.6	604.89	647.10	715.61	5524.3	556.69	568.00	608.57	4798.2
	RC	**547.44**	592.01	732.63	4926.2	**547.44**	550.17	555.64	6048.5	**547.44**	555.32	592.07	5458.7	594.62	638.40	682.75	5462.0	**547.44**	555.93	584.84	5588.7
Average		320.35	345.96	408.71	1750.1	319.30	322.91	329.88	2109.0	320.18	331.69	362.38	1815.9	336.52	361.75	396.06	1923.3	320.89	328.15	346.61	1824.0

Table 6. Number of *best run* on PHHPDP instances

De	*Type*	TS_C	$TS_{I\text{-}F\text{-}S}$	TS_{FLS}	TS_{ILS}	$TS_{F\text{-}I\text{-}P}$
50	*C*	14	36	21	17	22
	R	18	23	18	20	21
	RC	12	36	21	21	22
100	*C*	10	32	22	1	19
	R	26	52	19	17	36
	RC	16	32	25	6	24
350	*C*	21	29	29	0	27
	R	19	30	22	0	19
	RC	30	43	33	0	34
Total		166	313	210	82	224

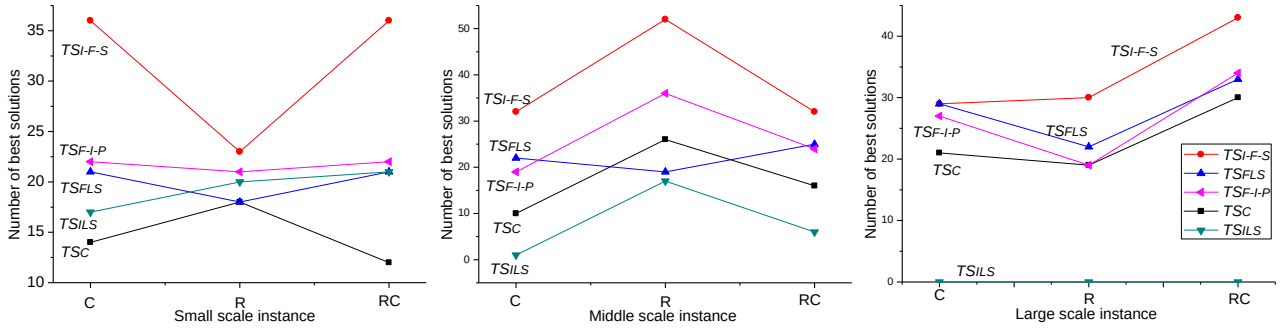

Figure 2. Number of *best run* for different scale-type instances

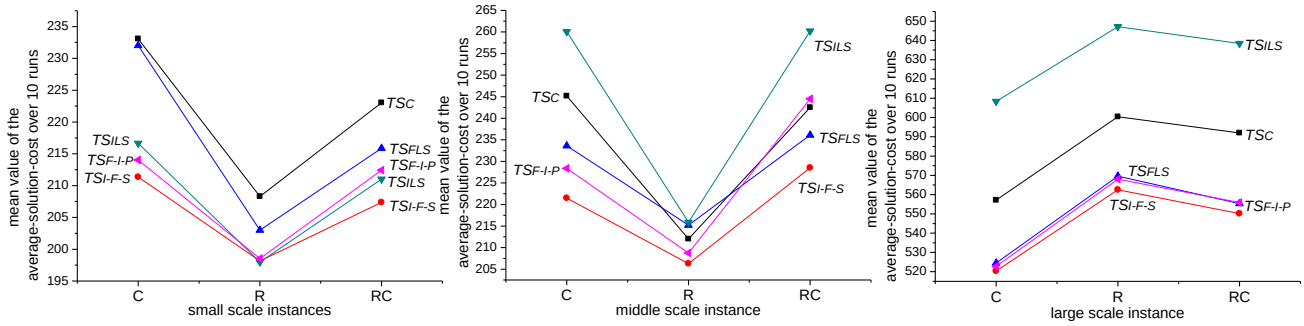

Figure 3. Mean average solution cost over different scale-type instances

In Figure 4, we show routes given by $TS_{I\text{-}F\text{-}S}$ for the VRPTW C104-based instance. This instance contains 30 clients and a horizon of 7 days. We show the routes of days 1 and 5. In Figure 4 each client is labeled with a client-number and its demand type. As stated in Section 5.1, all clients are P_1 patients. If a client is also a P_2 (or P_3) patient, we mark P_2 (or P_3) beside this client. Otherwise, this client is marked as P_1.

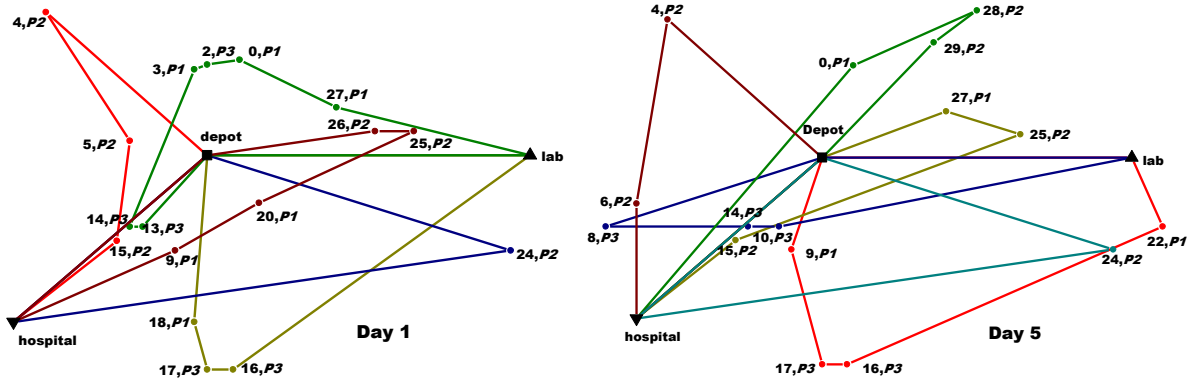

Figure 4. Routes of $TS_{I\text{-}F\text{-}S}$ for C104-based test instance

5.4 Real-World Case Study

In this section, we test the performances of our proposed algorithms using field data of OIKIA Company HHC operations, which is located in Saint-Etienne, a French city of about 450,000 inhabitants. OIKIA provides various levels of home health care depending on customer's need. We collected five sets of field data, corresponding to five different weeks

(5 working days a week). The customers are citizens of Saint Etienne. Two vehicles provide HHC services to these customers. Most customers need services two or three days a week. We use google map (http://maps.google.fr/) to get the travel distance between any two points, and assume a vehicle speed of 35 KM/h. Detailed field data, e.g., customer number in each data set, are shown in Table 7. We did not get access to the actual routes, but OIKIA's manager and planners gave their rules to assign and schedule the demands. In the current OIKIA HHC operations, the company has some predetermined rules to construct vehicle routes. The scheduler develops weekly vehicle scheduling according to these routes manually. The scheduling includes the following steps: (1) Try to balance the number of customers served each day, determine the visit day for each customer requiring multiple visits. At the end of this step, the visit requirements of each day are known. (2) For each day, the customers are divided into groups according to their geographic distribution, i.e., the zip codes. Each group is assigned to one vehicle (driver). (3) Calculate the workload of each vehicle and adjust the customers between vehicle routes with regards to the workload balancing among different vehicle drivers. For example, exchange and relocate customer between routes, especially the customers located on the border between different zip districts. We refer to these solutions executed by the company as the 'OIKIA solution'.

For each data instance, we have also run our $TS_{I\text{-}F\text{-}S}$ 10 times, each run with a computation time of 2 minutes. The costs of OIKIA and $TS_{I\text{-}F\text{-}S}$ solutions are presented and compared in Table 7. Columns '*vehicle1*' and '*vehicle2*' represent average daily travel distance of each vehicle, and column '*Max-route*' represents the travel distance of the longest route during a week (the optimization objective of this paper). The last column '*Dev*' indicates the percentage deviations of '*Max-route*' between OIKIA and our $TS_{I\text{-}F\text{-}S}$ solutions. Regarding the longest route during a week, clearly we get better result than that of the OIKIA solution. For five real-life instances, the longest route's distances of $TS_{I\text{-}F\text{-}S}$ is shorter than that of OIKIA with deviations of 10.08%, 16.59%, 15.73%, 8.41% and 9.31%. The $TS_{I\text{-}F\text{-}S}$ approach finds better solutions that balance the workloads of the two vehicles (drivers). Meanwhile, the superiority of $TS_{I\text{-}F\text{-}S}$ solution is also notable with regard to the total travel distances of each vehicle during a week, although it is not the original objective in our work. $TS_{I\text{-}F\text{-}S}$ solutions reduce the total travel distances of *vehicle* 1 and 2 by 2.15% and 6.88% respectively. For the real-world case study, we find that for some customers (about 30-40%), the visit days (i.e. the selected pattern) are different between 'OIKIA solution' and our result. Of course, almost for all the routes, their structures are different in two solutions. But some good components (a visit sequence of some customers) exist in routes of both solutions. The two main weaknesses of the OIKIA planner's procedure are: (1) they not take into account temporal time windows in the clustering step; (2) the clustering is based on a static geographical decomposition (ZIP codes) which does not depend on actual instance. Our integrated approach tends to avoid these defaults.

Table 7. Comparison between the OIKIA solutions and the $TS_{I\text{-}F\text{-}S}$ solutions

			OIKIA solution (KM)			$TS_{I\text{-}F\text{-}S}$ (KM)			
Instance	*n*	*K*	*vehicle 1*	*vehicle 2*	*Max-route*	*vehicle 1*	*vehicle 2*	*Max-route*	*Dev.%*
Real-1	58	2	96.1	92.9	102.7	92.0	91.7	93.3	-10.08
Real-2	54	2	90.7	99.0	106.8	86.5	87.4	91.6	-16.59
Real-3	50	2	83.5	90.2	98.6	83.8	84.2	85.2	-15.73
Real-4	46	2	88.0	94.2	98.0	87.4	87.8	90.4	-8.41
Real-5	49	2	91.2	97.3	103.3	90.5	92.0	94.5	-9.31
Average			89.9	94.7	101.9	88.0	88.6	91.0	-12.02

5.5 Test on min-max multiple TSP benchmarks

This section considers a special case with only P_1 patients without time window constraints over a planning horizon of one day. The problem reduces to the classical *Min-Max Multiple Traveling Salesman Problem* (*min-max* MTSP). We test $TS_{I\text{-}F\text{-}S}$ on the *min-max* MTSP benchmark, and compare our results with existing solutions. Golden et al. [40] proposed a very sophisticated tabu search based adaptive memory procedure to solve a class of VRPs with *min-max* objective, including the *min-max* MTSP. They generated *min-max* MTSP test instances and solved them. Somhom et al. [37] designed a competition-based neural network, denoted NN, for the *min-max* MTSP. Their results were still the best known results for these instances. They also generated new standard *min-max* MTSP test instances and gave the solutions by their neural network. Meanwhile, Somhom et al. [37] also solved the Golden's instances and compared the results.

We test our best approach $TS_{I\text{-}F\text{-}S}$ on these two sets of test instances, for each of which two stopping criteria are adopted for the $TS_{I\text{-}F\text{-}S}$. The first criterion is defined on the maximal iteration number of $TS_{I\text{-}F\text{-}S}$, i.e., N_1=15000, N_2=9000 and T=∞. Meanwhile, since Somhom's approach is rather fast and Somhom's running time is shorter than that of Golden's heuristic, we adopt maximal run time as the second stop criterion for $TS_{I\text{-}F\text{-}S}$. Since Somhom used a rather old computer (a Sun Sparc 10 with a CPU frequency of about 100 MHz), for each test instance, we set 10% of Somhom's computing time as the stop criterion of $TS_{I\text{-}F\text{-}S}$, i.e., N_1=∞, N_2=∞, and T is the one tenth of Somhom's computing time. In the following $TS_{I\text{-}F\text{-}S}$ under this set of parameters is denoted $TS_{I\text{-}F\text{-}S}'$. $TS_{I\text{-}F\text{-}S}$ algorithm with each stopping criterion runs 10 times for each instance.

Results of our algorithm are detailed in Tables 8 and 9. In Table 8, columns '*Instance*', '*n*' and '*K*' give the instance, the number of patients and vehicles. For each algorithm, the best solution over 10 runs (see columns '*Best*') and the average single run computing time in seconds (see columns '*CPU*') are given. Deviations between *NN* and our tabu search approach with the two stop criteria are given in columns '*Dev*%'.

Similarly, Table 9 compares $TS_{I\text{-}F\text{-}S}$ with the *ATS* approach of Golden et al. [40] and *NN*

approach of Somhom et al. [37] on Golden's test instances. For each Golden's instance, Golden et al. were able to generate the best known result. Note that in Table 9, columns '*Dev*' indicate the deviation between '*ATS*' and '$TS_{I\text{-}F\text{-}S}$' and the deviation between '*NN*' and '$TS_{I\text{-}F\text{-}S}'$', respectively.

Although our heuristics are not designed for the *min-max* MTSP, results in Tables 8 and 9 indicate that the performance of the proposed $TS_{I\text{-}F\text{-}S}$ algorithm is still very good. $TS_{I\text{-}F\text{-}S}$ with first stop criterion is able to improve the best known solution for 18 out of 24 Somhom's test instances. The average best solution cost found by $TS_{I\text{-}F\text{-}S}$ is 4.17% smaller than the previous best known solution. Even with reduced running time our approach, $TS_{I\text{-}F\text{-}S}'$, still can find better solutions for 14 out of 24 test instances. Regarding Golden's instances in Table 9, our $TS_{I\text{-}F\text{-}S}$ algorithm with the first stop criterion is able to get 5 new best solutions and find 6 existing best solutions out of 17 instances. But using the first stop criterion, $TS_{I\text{-}F\text{-}S}$ requires long computing times compared with ATS, which was running on a SUN SPARC 10. However, $TS_{I\text{-}F\text{-}S}'$ allows getting solutions close to best ones, as the average deviation from the best solution of the literature is 1.6%. Furthermore, $TS_{I\text{-}F\text{-}S}'$ obviously outperforms the method of Somhom et al. [37]. $TS_{I\text{-}F\text{-}S}'$ is able to improve Somhom's best solutions for 14 out of 17 instances and gives two new best solutions. The average best solution cost found by $TS_{I\text{-}F\text{-}S}'$ is 1.37% smaller than previous Somhom's best solution. Despite the bias in the comparison between algorithms running on different machines, we can state that our approach gets slightly worse on average than the state of the art heuristic (ATS) on min-max MTSP, and outperforms another very fast heuristic (NN).

Table 8. Results on Somhom min-max MTSP test instances

			NN		$TS_{I\text{-}F\text{-}S}'$			$TS_{I\text{-}F\text{-}S}$		
Instance	*N*	*K*	*Best*	*CPU*[a]	*Best*	*CPU*	*Dev%*	*Best*	*CPU*	*Dev%*
eil22	22	2	**157**	0.3	159	0.0	1.3	159	2	1.3
		3	117	0.3	**115***	0.0	-1.7	115*	2	-1.7
		4	111	0.2	**109***	0.0	-1.8	109*	2	-1.8
eil30	30	2	230	0.6	**224***	0.1	-2.7	224*	6	-2.7
		3	174	0.4	**165***	0.0	-5.5	165*	4	-5.5
		4	171	0.4	**156***	0.0	-9.6	156*	4	-9.6
eil51	51	2	247	1.9	**224**	0.2	-10.3	222*	29	-11.3
		3	170	1.9	**161**	0.2	-5.6	157*	24	-8.3
		4	136	2.0	**126***	0.2	-7.9	126*	23	-7.9
eil76	76	2	289	4.8	**281**	0.5	-2.9	275*	97	-5.1
		3	205	4.9	205	0.5	0.0	192*	62	-6.8
		4	**159**	5.1	174	0.5	8.6	155*	51	-2.6
kroA100	100	2	**11484**	17.3	12077	1.7	4.9	11525	188	0.4
		3	9062	15.9	**8783**	1.6	-3.2	8054*	147	-12.5
		4	7497	14.8	**7079**	1.5	-5.9	6729*	127	-11.4

kroA150	150	2	14885	24.5	**14844**	2.5	-0.3	13960*	920	-6.6
		3	**10527**	23.1	11658	2.3	9.7	10768	485	2.2
		4	**8571**	22.7	9080	2.3	5.6	8147*	412	-5.2
kroA200	200	2	**17353**	37.3	17507	3.7	0.9	16045*	2138	-8.2
		3	**11502**	37.9	12765	3.8	9.9	12359	1354	6.9
		4	10433	33.3	**10413**	3.3	-0.2	9943*	851	-4.9
eil101	101	2	340	15.1	**339**	1.2	-0.3	337*	298	-0.9
		3	**232**	13.6	241	1.4	3.7	237	179	2.1
		4	**187**	15.5	188	1.6	0.5	187	139	0.0
Average				12.2		1.2	-0.5		314.4	-4.2

[a] CPU times obtained on a SUN SPARC 10;

Asterisk indicates the new best solution and boldface shows best result between $TS_{I\text{-}F\text{-}S}'$ and *NN*.

Table 9. Results on Golden min-max MTSP test instances

			ATS		$TS_{I\text{-}F\text{-}S}$				*NN*		$TS_{I\text{-}F\text{-}S}'$		
Instance	*n*	*K*	*Best*	*CPU*[a]	*Best*	*CPU*	*Dev%*		*Best*	*CPU*	*Best*	*CPU*	*Dev%*
CMT_50	50	5	110.20	210	110.17*	19.6	0.0		112.70	3.5	**110.17***	0.4	-2.3
		6	99.26	190	99.18*	18.9	-0.1		102.23	5.9	**99.18***	0.6	-3.1
		7	91.62	160	91.62	19.5	0.0		94.34	10.2	**91.62**	1.0	-3.0
CMT_75	75	10	91.21	210	91.22	54.4	0.0		94.14	5.3	**92.52**	0.5	-1.8
		11	88.72	210	89.09	57.4	0.4		93.84	10.7	**92.18**	1.1	-1.8
		12	88.08	210	88.08	52.0	0.0		90.80	13.3	**88.64**	1.3	-2.4
CMT100	100	8	111.12	610	110.81*	116.6	-0.3		**115.18**	11.5	115.66	1.2	0.4
		9	105.39	610	105.48	113.3	0.1		**107.34**	17.5	109.59	1.7	2.1
		10	100.37	550	101.94	115.6	1.5		105.68	25.7	**103.33**	2.6	-2.3
CMT150	150	12	100.12	1100	99.99*	318.3	-0.1		**104.30**	36.5	106.60	3.7	2.2
CMT199	199	15	99.86	51	99.86	825.7	0.0		103.87	40.4	**103.22**	4.0	-0.6
CMT120	120	7	199.39	1400	199.62	170.2	0.1		202.71	24.6	**201.73**	2.5	-0.5
CMT12_100	100	10	117.05	7	117.05	101.7	0.0		117.05	6.1	117.05	0.6	0.0
Fisher_71	4	4	65.10	960	65.08	59.3	0.0		65.46	4.5	**65.11**	0.5	-0.5
	5	5	59.32	790	59.32	36.6	0.0		61.79	6.1	**59.74**	0.6	-3.4
	6	6	55.19	700	55.10*	44.3	-0.2		58.25	9.0	**55.10***	0.9	-5.7
Fisher_134	134	7	293.54	41	293.54	250.4	0.0		296.02	20.4	**294.44**	2.0	-0.5
Average				471.1		139.6	0.1			14.8		1.5	-1.4

[a] CPU times obtained on a SUN SPARC 10;

Asterisk indicates the new best solution and boldface shows best result between $TS_{I\text{-}F\text{-}S}'$ and *NN*.

5.6 Test on the Periodic Traveling Salesman Problem benchmarks

Besides the min-max objective adopted in this paper, some other objectives, such as total travel distance minimization, are widely used in VRPs. Our TS algorithm is flexible and can be extended to other objectives. In this subsection, we simply explain how to extend our

heuristic to minimize the total travel distance.

To minimize the total travel distance, we keep the initial solution of Section 4.1 despite its poor performance for this new objective. During the neighbor search, solutions are evaluated as $f'(s) = c'(s)$, where $c'(s)$ is the total travel distance. Similar evaluation is needed in the local search procedure for intra-route improvement. Finally, the aspiration value of each attribute is the total travel distances of the best feasible solution found with this attribute.

The modified $TS_{I\text{-}F\text{-}S}$ is tested on existing *Periodic Traveling Salesman Problem* (PTSP) benchmarks. The PTSP can be seen as a special case of our problem with only P_1 patients and only one vehicle on each day. Compared with *min-max* MTSP solved in Section 5.3, the PTSP has been extensively studied [19, 64-67]. Note that the classical PTSP has a constraint that at least one customer must be visited each day, which was introduced by Chao et al. [65]. This constraint is not meaningful in our problem and most real-life applications, so it is ignored in our heuristic and experiments. In order to test our approach, first, based on the classical PTSP benchmarks of Cordeau et al. [6], we derive 22 new test instances. We set new patterns to the customers in the Cordeau's benchmark, which forbid empty routes on each day in feasible solutions. Other data, e.g., the locations of the depot and customers are used directly. $TS_{I\text{-}F\text{-}S}$ is compared with the tabu search designed by Cordeau et al. [6] for the PTSP. For each test instance Cordeau's tabu search is executed 10 times on a 2.66G CPU, and each run stops with maximal 15000 iterations. Our $TS_{I\text{-}F\text{-}S}$ also solves each instance 10 times. It is stopped when running time reached the same as Cordeau's tabu search. Results are shown in Table 10. Table 10 gives best solution costs of both Cordeau's algorithm and $TS_{I\text{-}F\text{-}S}$, their running time in second and deviations, where Column '*t*' gives the number of days in the planning horizon. $TS_{I\text{-}F\text{-}S}$ is competitive with respect to Cordeau's algorithm. $TS_{I\text{-}F\text{-}S}$ is able to find 6 better solutions out of 22 test instances and the same best solutions for 8 instances. For the 8 remaining instances, Cordeau's approach outperforms $TS_{I\text{-}F\text{-}S}$ algorithm. The superiority and difference between $TS_{I\text{-}F\text{-}S}$ and Cordeau's algorithm are not clear. Both approaches use similar schemes and our approach wastes time in infeasible local search designed to deal with constraints that do not exist in PTSP that does not have time windows and precedence constraints.

Meanwhile, we find PTSP computational experiments of Cacchiani et al. [27] also relax the constraint of at least one customer visited every day. We compare our $TS_{I\text{-}F\text{-}S}$ with the set-covering based heuristic of Cacchiani et al. on series of classical PTSP instances. Cacchiani et al. executed their heuristic on an Intel Xeon 2.67 GHz CPU for solving each PTSP instance, with a time limit of 2 hours. We used $TS_{I\text{-}F\text{-}S}$ to solve each instance 10 times. For small instances (less than 100 customers), we set the stop criterion as the total running time of 5 minutes, and for larger instances the $TS_{I\text{-}F\text{-}S}$ was executed with time limit of 15 minutes. The comparison is presented in Table 11. To have a full comparison with all other

PTSP methods, we also present in Table 11 results of the following state-of-the-art algorithms: the heuristics of Chao et al. [65] in column '*CGW*' and of Paletta [66] in '*P*', the tabu search of Cordeau et al. [6] in '*CGL*', the heuristic algorithms of Bertazzi et al. [67] in '*BPS*' and of Hemmelmayr et al. [19] in '*HDH*', and a set-covering based heuristic algorithm of Cacchiani et al. [27] in '*CHT*'. Since both Cacchiani et al. [27] and our $TS_{I\text{-}F\text{-}S}$ neglect the constraint introduced by Chao et al. [65], results of these two approaches may not be feasible with respect to this constraint. Such results are marked with double asterisks '**'. We consider all instances in the computation of the average percentage gap with approach of Cacchiani et al. For other state-of-the-art algorithms, we do not consider these 'infeasible' instances when calculating percentage gaps. As shown in Table 11, *CHT* finds 3 new best solutions which break Chao et al. constraint (p03, p06 and p09), while $TS_{I\text{-}F\text{-}S}$ finds 7 such solutions (p01, p03, p04, p06, p09, p17 and p19). For three common instances p03, p06 and p09, $TS_{I\text{-}F\text{-}S}$ finds better solutions than *CHT*. Besides these new solutions, $TS_{I\text{-}F\text{-}S}$ reaches the same best solutions 5 times (instances p11-p15).

Although our approach is able to find new best solutions, it is slightly dominated on average by several specialized algorithms; the gaps among other algorithms are: -0.96%, 0.78%, 0.26%, 0.73%, 1.95% and 1.44%. Such results cannot completely prove the benefit of $TS_{I\text{-}F\text{-}S}$ for solving the classical PTSP when compared with specialized state-of-the-art algorithms. However, we think that the results on PTSP are acceptable for an approach which includes many useless features for this problem. Moreover, algorithms considering explicitly the Chao et al. constraint are favored on some instances. The Chao et al. constraint requires all vehicles to be used. It reduces the solution space and can be easily handled in many solution algorithms. $TS_{I\text{-}F\text{-}S}$ and *CHT* are therefore penalized, when the best (near optimum) solutions of the PTSP without Chao et al. constraint satisfy this constraint. In such cases, both approaches have to explore a larger solution space. The comparison between *CHT* and $TS_{I\text{-}F\text{-}S}$ shows that $TS_{I\text{-}F\text{-}S}$ outperforms *CHT* when best solutions found do not satisfy Chao et al. constraint, and *CHT* outperforms $TS_{I\text{-}F\text{-}S}$ in the opposite case. In other words, we state that $TS_{I\text{-}F\text{-}S}$ performs better than *CHT* when best solutions do not use all vehicles, and it performs worse otherwise. It means that $TS_{I\text{-}F\text{-}S}$ can be improved when it is likely that not all vehicles will be used.

Table 10. Results on New Periodic Traveling Salesman Problem test instances

Instance	*n*	*t*	*Cordeau*	$TS_{I\text{-}F\text{-}S}$	*CPU*	*Dev%*
t-p1-new	50	2	552.28	**551.46**	4.8	-0.15
t-p2-new	50	5	1129.80	**1127.41**	5.4	-0.21
t-p3-new	50	5	**590.58**	**590.58**	3.6	0.00
t-p4-new	75	2	**591.03**	593.92	9.0	0.49
t-p5-new	75	5	**1387.62**	1394.73	9.6	0.51
t-p6-new	75	10	**817.87**	849.31	7.2	3.70

t-p7-new	100	2	690.90	**688.26**	14.4	-0.38
t-p8-new	100	5	**1621.08**	1676.45	15.0	3.30
t-p9-new	100	8	977.58	**964.31**	9.6	-1.38
t-p10-new	100	5	**1373.60**	1398.28	13.8	1.77
t-p11-new	65	4	**490.97**	**490.97**	6.0	0.00
t-p12-new	87	4	**664.10**	**664.10**	9.0	0.00
t-p13-new	109	4	**830.80**	**830.80**	13.2	0.00
t-p14-new	131	4	**994.60**	**994.60**	18.0	0.00
t-p15-new	153	4	**1157.07**	**1157.07**	23.4	0.00
t-p16-new	48	4	742.90	**722.82**	4.2	-2.78
t-p17-new	66	4	**918.44**	**918.44**	6.6	0.00
t-p18-new	84	4	**935.13**	**935.13**	10.2	0.00
t-p19-new	102	4	**1079.30**	1097.58	13.8	1.67
t-p20-new	120	4	1206.45	**1198.96**	15.6	-0.62
t-p21-new	77	4	**1396.18**	1396.77	7.8	0.04
t-p22-new	154	4	**4326.08**	4375.27	24.0	1.12
Average						0.32

Boldface shows better result between two approaches.

Table 11. Results on classical Periodic Traveling Salesman Problem test instances

Instance	*CGW*	*P*	*CGL*	*BPS*	*HDH*	*CHT*	$TS_{I\text{-}F\text{-}S}$
p01	442.1	436.50	439.02	436.50	432.10	432.10	**428.98****
p02	1106.7	1122.44	1111.93	1122.44	1106.84	**1105.81**	1111.93
p03	474.0	469.16	469.69	469.64	467.42	446.17****	**428.98****
p04	554.2	559.68	556.21	559.49	552.39	550.07	**547.24****
p05	1394.0	1387.90	1389.54	1384.75	1384.58	**1384.15**	1384.58
p06	657.3	643.59	651.28	655.06	652.65	581.94****	**556.82****
p07	662.4	—	660.41	**646.65**	649.17	658.09	657.89
p08	1635.2	—	1634.68	1633.92	1615.51	**1612.60**	1624.58
p09	735.3	—	734.16	733.13	729.33	698.04**	**660.54****
p10	1248.8	—	1240.01	1249.15	**1237.72**	1239.96	1245.71
p11	**491.0**	**490.97**	**490.97**	**490.97**	**490.97**	**490.97**	**490.97**
p12	**664.1**	**664.10**	**664.10**	**664.10**	**664.10**	**664.10**	**664.10**
p13	**830.8**	**830.80**	**830.80**	**830.80**	**830.80**	**830.80**	**830.80**
p14	**994.6**	**994.60**	**994.60**	**994.60**	**994.60**	**994.60**	**994.60**
p15	**1157.1**	**1157.07**	**1157.07**	**1157.07**	**1157.07**	1157.12	**1157.07**
p16	726.8	660.12	660.12	660.12	660.12	**649.96**	662.28
p17	776.5	776.43	776.43	776.43	776.71	774.54	**764.49****
p18	**873.7**	876.44	**873.73**	876.44	875.82	887.05	887.05
p19	974.6	958.51	958.88	958.51	965.54	974.60	**939.35****
p20	1053.6	**1033.58**	1034.51	**1033.58**	1035.51	1053.59	1077.85
p21	1379.1	—	1375.08	**1375.07**	**1375.07**	1375.08	1375.08

p22	4323.6	–	4319.72	4323.49	**4312.31**	4312.32	4318.07
p23	8753.3	8390.53	8553.10	8498.00	**8349.26**	8405.10	8554.91
pr01	–	**2064.84**	2068.46	**2064.84**	**2064.84**	**2064.84**	2076.89
pr02	–	3232.72	3293.50	3231.50	3208.49	**3208.22**	3317.17
pr03	–	4084.75	4106.72	4118.63	**4045.73**	4065.15	4120.76
pr04	–	4636.67	4661.97	4621.36	**4547.77**	4557.92	4689.63
pr05	–	4757.90	4698.83	4682.54	4628.24	**4623.86**	4707.66
pr06	–	5688.42	5699.96	5595.45	**5529.68**	5559.11	5699.84
pr07	–	4479.65	4453.15	4474.17	**4436.31**	4446.60	4458.21
pr08	–	–	5405.40	5475.70	**5370.59**	5383.44	5475.72
pr09	–	7405.52	7469.73	7346.32	**7244.02**	7256.65	7464.23
pr10	–	8394.52	8493.74	8415.31	**8216.48**	8243.32	8492.69
Dev%	-0.96	0.78	0.26	0.73	1.95	1.44	

Boldface shows the best result among approaches.

6 Conclusions and future research

This paper investigates a special periodic vehicle routing problem with time windows in home health care industry, an extension of the classical PVRP and PTSP. The problem is of interest because of its theoretical complexity and of the important practical applications in the home health care logistics. We propose hybridization of tabu search and different local search schemes, for solving this complex problem. One salient feature of our approach is the hybridization of feasible and infeasible local search methods in the tabu search algorithm for solving the vehicle routing problem. Different integration strategies of feasible and infeasible local searches are tested on different scales and types test instances. We find experimentally that infeasible local search with small probability followed by a feasible local search with high probability outperforms other strategies. Our proposed methods are also extensively tested on different test instances, including VRPTW-based benchmarks, min-max Multiple TSP, Periodic Traveling Salesman Problem, and real-life data from a HHC company.

Future research can be pursued in several directions. First, it is interesting to consider explicitly the planning decision and daily routing decisions instead of the implicit patient demand pattern model of this paper. This requires the modeling of patient demand and the therapeutic protocols of patients. Another important research direction is to take into account the uncertainties of demand and availabilities times in the planning and routing decisions. Another relevant research direction is the real time routing decisions to face arrival of emergency demands and random perturbations.

Acknowledgement

This project is partly supported by Saint Etienne Metropole, the labex IMOBS3, and National Natural Science Foundation of China (NSFC) under Grant 71131005.

References

[1] Cordeau JF, Laporte G, Mercier A. A unified tabu search heuristic for vehicle routing problems with time windows. Journal of the Operational Research Society. 2001;52:928-36.

[2] Bektas T. The multiple traveling salesman problem: an overview of formulations and solution procedures. Omega. 2006;34:209-19.

[3] Carter AE, Ragsdale CT. Scheduling pre-printed newspaper advertising inserts using genetic algorithms. Omega. 2002;30:415-21.

[4] Gendreau M, Hertz A, Laporte G, Stan M. A Generalized Insertion Heuristic for the Traveling Salesman Problem with Time Windows. Operations Research. 1998;46:330-5.

[5] Savelsbergh MWP. Local search in routing problems with time windows. Annals of Operations Research. 1985;4:285-305.

[6] Cordeau JF, Gendreau M, Laporte G. A tabu search heuristic for periodic and multi-depot vehicle routing problems. Networks. 1997;30:105-19.

[7] Begur SV, Miller DM, Weaver JR. An integrated spatial DSS for scheduling and routing home-health-care nurses. Interfaces. 1997;27:35-48.

[8] Cheng E, Rich JL. A Home Health Care Routing and Scheduling Problem. Technical Report, Rice University, Houston, Texas, USA. 1998.

[9] Bertels S, Fahle T. A hybrid setup for a hybrid scenario: combining heuristics for the home health care problem. Computers & Operations Research. 2006;33:2866-90.

[10] Eveborn P, Flisberg P, Rönnqvist M. Laps Care—an operational system for staff planning of home care. European Journal of Operational Research. 2006;171:962-76.

[11] Kergosien Y, Lenté C, Billaut JC. Home health care problem: An extended multiple traveling salesman problem. Multidisciplinary International Conference on Scheduling : Theory and Applications. Dublin, Ireland2009. p. 10-2.

[12] Trautsamwieser A, Gronalt M, Hirsch P. Securing home health care in times of natural disasters. OR Spectrum. 2011;33:787-813.

[13] Nickel S, Schröder M, Steeg J. Mid-term and short-term planning support for home health care services. European Journal of Operational Research. 2012;219:574-87.

[14] Beltrami EJ, Bodin LD. Networks and vehicle routing for municipal waste collection. Networks. 1974;4:65-94.

[15] Russell R, Igo W. An assignment routing problem. Networks. 1979;9:1-17.

[16] Christofides N, Beasley JE. The period routing problem. Networks. 1984;14:237-56.

[17] Chao I, Golden BL, Wasil E. An improved heuristic for the period vehicle routing problem. Networks. 1995;26:25-44.

[18] Mourgaya M, Vanderbeck F. Column generation based heuristic for tactical planning in multi-period vehicle routing. European Journal of Operational Research. 2007;183:1028-41.

[19] Hemmelmayr VC, Doerner KF, Hartl RF. A variable neighborhood search heuristic for periodic routing problems. European Journal of Operational Research. 2009;195:791-802.

[20] Pirkwieser S, Raidl GR. Boosting a Variable Neighborhood Search for the Periodic Vehicle Routing Problem with Time Windows by ILP Techniques. Proceedings of the 8th Metaheuristic International Conference (MIC 2009), Hamburg, Germany, July 2009.

[21] Pirkwieser S, Raidl GR. Multilevel Variable Neighborhood Search for Periodic Routing Problems. In: Cowling P, Merz P, editors. Evolutionary Computation in Combinatorial Optimization: Springer Berlin / Heidelberg; 2010:226-38.

[22] Pirkwieser S, Raidl GR. A Column Generation Approach for the Periodic Vehicle Routing Problem with Time Windows. Proceedings of the International Network Optimization Conference 2009, Pisa, Italy, April 2009, 26-29.

[23] Pirkwieser S, Raidl GR. Matheuristics for the periodic vehicle routing problem with time windows. Proceedings of Matheuristics. 2010: third international workshop on model-based metaheuristics, Vienna, Austria, June 2010, 28-30.

[24] Gulczynski D, Golden B, Wasil E. The period vehicle routing problem: New heuristics and real-world variants. Transportation Research Part E: Logistics and Transportation Review. 2011;47:648-68.

[25] Vidal T, Crainic TG, Gendreau M, Lahrichi N, Rei W. A Hybrid Genetic Algorithm for Multidepot and Periodic Vehicle Routing Problems. Operations Research. 2012;60:611-24.

[26] Vidal T, Crainic TG, Gendreau M, Prins C. A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. Computers & Operations Research. 2013;40:475-89.

[27] Cacchiani V, Hemmelmayr VC, Tricoire F. A set-covering based heuristic algorithm for the periodic vehicle routing problem. Discrete Applied Mathematics. 2014;163, Part 1:53-64.

[28] Baldacci R, Bartolini E, Mingozzi A, Valletta A. An exact algorithm for the period routing problem. Operations Research. 2011;59:228-41.

[29] Lacomme P, Prins C, Ramdane-Chérif W. Evolutionary algorithms for periodic arc routing problems. European Journal of Operational Research. 2005;165:535-53.

[30] Cornillier F, Boctor FF, Laporte G, Renaud J. A heuristic for the multi-period petrol station replenishment problem. European Journal of Operational Research. 2008;191:295-305.

[31] Angelelli E, Bianchessi N, Mansini R, Speranza MG. Short Term Strategies for a Dynamic Multi-Period Routing Problem. Transportation Research Part C: Emerging Technologies. 2009;17:106-19.

[32] Wen M, Cordeau JF, Laporte G, Larsen J. The dynamic multi-period vehicle routing problem. Computers & Operations Research. 2010;37:1615-23.

[33] Angelelli E, Grazia Speranza M. The periodic vehicle routing problem with intermediate facilities. European Journal of Operational Research. 2002;137:233-47.

[34] Francis P, Smilowitz K, Tzur M. The period vehicle routing problem with service choice. Transportation Science. 2006;40:439-54.

[35] Yu B, Yang ZZ. An ant colony optimization model: The period vehicle routing problem with time windows. Transportation Research Part E: Logistics and Transportation Review. 2011;47:166-81.

[36] França PM, Gendreau M, Laporte G, Müller FM. The m-traveling salesman problem with minmax objective. Transportation Science. 1995;29:267-75.

[37] Somhom S, Modares A, Enkawa T. Competition-based neural network for the multiple travelling salesmen problem with minmax objective. Computers & Operations Research. 1999;26:395-407.

[38] Modares A, Somhom S, Enkawa T. A self‐organizing neural network approach for multiple traveling salesman and vehicle routing problems. International Transactions in Operational Research. 1999;6:591-606.

[39] Arkin EM, Hassin R, Levin A. Approximations for minimum and min-max vehicle routing problems. Journal of Algorithms. 2006;59:1-18.

[40] Golden BL, Laporte G, Taillard ÉD. An adaptive memory heuristic for a class of vehicle routing problems with minmax objective. Computers & Operations Research. 1997;24:445-52.

[41] Valle CA, Martinez LC, da Cunha AS, Mateus GR. Heuristic and exact algorithms for a min–max selective vehicle routing problem. Computers & Operations Research. 2011;38:1054-65.

[42] Dumas Y, Desrosiers J, Soumis F. The pickup and delivery problem with time windows. European Journal of Operational Research. 1991;54:7-22.

[43] Savelsbergh M, Sol M. DRIVE: Dynamic routing of independent vehicles. Operations Research. 1998;46:474-90.

[44] Bard JF, Jarrah AI. Integrating commercial and residential pickup and delivery networks: A case study. Omega. 2013;41:706-20.

[45] Bent R, Hentenryck PV. A two-stage hybrid algorithm for pickup and delivery vehicle routing problems with time windows. Computers & Operations Research. 2006;33:875-93.

[46] Ropke S, Cordeau JF, Laporte G. Models and branch‐and‐cut algorithms for pickup and delivery problems with time windows. Networks. 2007;49:258-72.

[47] Ropke S, Cordeau JF. Branch and cut and price for the pickup and delivery problem with time windows. Transportation Science. 2009;43:267-86.

[48] Gribkovskaia I, Laporte G, Shlopak A. A Tabu Search Heuristic for a Routing Problem Arising in Servicing of Offshore Oil and Gas Platforms. The Journal of the Operational Research Society. 2008;59:1449-59.

[49] Qian F, Gribkovskaia I, Laporte G, Halskau sr Ø. Passenger and pilot risk minimization in offshore helicopter transportation. Omega. 2012;40:584-93.

[50] Ma H, Cheang B, Lim A, Zhang L, Zhu Y. An investigation into the vehicle routing problem with time windows and link capacity constraints. Omega. 2012;40:336-47.

[51] Yu B, Yang ZZ, Yao B. An improved ant colony optimization for vehicle routing problem. European Journal of Operational Research. 2009;196:171-6.

[52] Jozefowiez N, Semet F, Talbi E-G. An evolutionary algorithm for the vehicle routing problem with route balancing. European Journal of Operational Research. 2009;195:761-9.

[53] Reimann M, Doerner K, Hartl RF. D-ants: Savings based ants divide and conquer the vehicle routing problem. Computers & Operations Research. 2004;31:563-91.

[54] Prins C. A simple and effective evolutionary algorithm for the vehicle routing problem. Computers & Operations Research. 2004;31:1985-2002.

[55] Gajpal Y, Abad P. An ant colony system (ACS) for vehicle routing problem with simultaneous delivery and pickup. Computers & Operations Research. 2009;36:3215-23.

[56] Della Croce F, Garaix T, Grosso A. Iterated local search and very large neighborhoods for the parallel-machines total tardiness problem. Computers & Operations Research. 2012;39:1213-7.

[57] Brandão J. A tabu search algorithm for the open vehicle routing problem. European Journal of Operational Research. 2004;157:552-64.

[58] Gendreau M, Hertz A, Laporte G. New insertion and postoptimization procedures for the traveling salesman problem. Operations Research. 1992;40:1086-94.

[59] Braysy O, Gendreau M. Vehicle routing problem with time windows, Part I: Route construction and local search algorithms. Transportation Science. 2005;39:104-18.

[60] Pedamallu CS, Ozdamar L. Investigating a hybrid simulated annealing and local search algorithm for constrained optimization. European Journal of Operational Research. 2008;185:1230-45.

[61] Tseng LY, Lin YT. A hybrid genetic local search algorithm for the permutation flowshop scheduling problem. European Journal of Operational Research. 2009;198:84-92.

[62] Solomon MM. Algorithms for the vehicle routing and scheduling problems with time window constraints. Operations Research. 1987;35:254-65.

[63] Gehring H, Homberger J. A parallel hybrid evolutionary metaheuristic for the vehicle routing problem with time windows. 1999.

[64] Paletta G. A multiperiod traveling salesman problem: Heuristic algorithms. Computers & Operations Research. 1992;19:789-95.

[65] Chao IM, Golden BL, Wasil EA. A new heuristic for the period traveling salesman problem. Computers & Operations Research. 1995;22:553-65.

[66] Giuseppe P. The period traveling salesman problem: a new heuristic algorithm. Computers & Operations Research. 2002;29:1343-52.

[67] Bertazzi L, Paletta G, Speranza MG. An improved heuristic for the period traveling salesman problem. Computers & Operations Research. 2004;31:1215-22.

Appendix

This appendix gives detailed computational results for VRPTW-based test instances. In these tables, '*Instance*' refers to the instance label, '*Best*', '*AVG*' and '*Worst*' are the best, the average and the worst solution cost of the 10 independent runs for each instance, '*BR*' is the number of best runs, '*CPU*' is the computational time in seconds.

Table 12. Detailed results on comparison of different p_{ILS} settings in $TS_{I\text{-}F\text{-}S}$

	p_{FLS}=80% and p_{ILS}=0%				p_{FLS}=80% and p_{ILS}=2%				p_{FLS}=80% and p_{ILS}=5%				p_{FLS}=80% and p_{ILS}=10%			
Instance	*Best*	*Average*	*Worst*	*BR*	*Best*	*Average*	*Worst*	*BR*	*Best*	*Average*	*Worst*	*BR*	*Best*	*Average*	*Worst*	*BR*
C101_30	214.65	284.56	526.75	0	207.06	210.13	233.24	0	205.75	208.18	214.33	2	207.06	208.00	214.65	0
C104_30	190.36	200.88	232.68	0	175.62	194.68	221.71	2	175.62	188.01	201.44	1	175.62	191.08	205.58	1
C109_60	223.39	234.81	252.52	3	223.39	223.39	223.39	10	223.39	223.39	223.39	10	223.39	225.68	246.26	9
C201_60	220.26	248.40	284.76	0	215.20	219.08	223.70	0	215.32	223.19	228.24	0	218.38	223.60	228.93	0
R108_30	188.49	206.04	221.72	1	188.49	190.26	206.14	9	188.49	188.50	188.50	5	188.49	189.90	202.58	5
R207_30	231.01	239.58	245.66	0	234.83	236.09	245.23	0	232.06	235.68	241.28	0	230.94	235.45	243.07	1
R210_60	208.84	230.13	246.23	1	208.84	211.56	215.63	6	208.84	211.56	215.63	6	208.84	210.52	218.80	8
RC105_60	197.59	220.01	256.87	0	193.09	194.50	197.85	4	193.09	194.48	197.03	1	193.44	194.99	199.04	0
RC201_60	214.22	243.01	276.22	1	229.62	233.26	239.21	0	236.62	240.37	244.53	0	237.68	241.12	244.86	0
RC204_60	228.22	231.14	250.01	8	228.22	229.66	242.62	9	228.22	228.22	228.22	10	228.22	230.23	248.30	9
	p_{FLS}=80% and p_{ILS}=20%				p_{FLS}=80% and p_{ILS}=30%				p_{FLS}=80% and p_{ILS}=40%				p_{FLS}=80% and p_{ILS}=50%			
Instance	*Best*	*Average*	*Worst*	*BR*	*Best*	*Average*	*Worst*	*BR*	*Best*	*Average*	*Worst*	*BR*	*Best*	*Average*	*Worst*	*BR*
C101_30	207.14	209.12	219.24	0	207.06	209.60	229.24	0	207.14	209.48	229.24	0	207.14	212.88	233.24	0
C104_30	175.62	195.38	205.92	1	175.62	194.23	208.92	1	175.62	192.82	214.23	1	183.67	193.95	202.15	0
C109_60	223.39	225.15	241.02	9	223.39	223.63	225.15	8	223.39	227.44	246.26	8	223.39	225.99	241.02	6
C201_60	215.32	222.75	241.37	0	215.25	224.10	241.43	0	216.39	231.19	247.89	0	213.32	234.87	256.97	1
R108_30	188.49	193.57	239.18	6	188.49	193.89	241.95	3	188.49	196.86	239.77	2	188.49	188.49	188.50	7
R207_30	232.03	235.68	243.41	0	230.94	234.57	235.66	1	234.83	235.39	238.37	0	230.94	235.07	238.37	1
R210_60	208.84	212.91	215.63	4	208.84	214.95	215.63	3	208.84	214.90	220.45	2	208.84	214.53	215.63	1
RC105_60	196.49	197.75	202.45	0	196.49	199.95	205.56	0	199.36	209.28	221.20	0	197.81	217.13	285.52	0
RC201_60	229.03	238.42	245.70	0	229.62	235.74	239.80	0	228.97	240.74	250.69	0	229.59	242.79	251.97	0
RC204_60	228.22	230.89	248.30	8	228.22	234.38	268.03	8	228.22	228.22	228.22	10	228.22	231.06	241.44	7

Table 13. Detailed results on small scale VRPTW-based instances

		TS_C					$TS_{I\text{-}F\text{-}S}$					TS_{FLS}					TS_{ILS}					$TS_{F\text{-}I\text{-}P}$				
Instance	*De*	*Best*	*Avg*	*Worst*	*N*	*CPU*	*Best*	*Avg*	*Worst*	*N*	*CPU*	*Best*	*Avg*	*Worst*	*N*	*CPU*	*Best*	*Avg*	*Worst*	*N*	*CPU*	*Best*	*Avg*	*Worst*	*N*	*CPU*
C101	50	214.33	290.60	461.07	0	55.6	205.75	212.08	233.24	1	62.2	214.65	284.56	526.75	0	71.1	207.14	212.42	231.85	0	53.5	207.14	217.73	240.16	0	57.4
C104	50	175.62	194.85	212.49	1	48.9	175.62	190.55	202.04	1	56.7	190.36	200.88	232.68	0	51.9	175.62	189.32	202.04	1	58.0	175.62	192.44	202.04	1	48.2
C109	50	220.53	223.78	253.04	9	51.6	220.53	220.53	220.53	10	53.3	220.53	220.98	225.02	9	53.5	220.53	220.58	221.00	9	57.4	220.53	223.74	252.61	9	54.3
C201	50	218.35	255.00	384.00	1	55.7	218.35	218.35	218.35	10	58.4	218.35	246.01	352.23	3	65.9	220.85	247.17	331.17	0	58.4	218.35	221.61	242.76	4	48.3
C203	50	229.74	235.07	244.73	1	52.9	229.74	230.63	238.63	9	45.0	229.74	239.61	296.80	2	47.7	229.74	233.41	242.67	5	51.3	229.74	230.37	233.63	5	59.3
C208	50	194.77	199.21	227.38	2	49.6	194.77	196.00	200.82	5	58.7	194.77	200.21	245.31	7	57.7	194.77	197.10	203.28	2	47.1	194.77	198.37	209.71	3	51.4
R101	50	181.10	190.48	203.54	4	44.9	181.10	187.79	207.49	2	40.4	181.10	194.35	207.49	1	46.0	181.10	182.66	194.10	2	59.5	181.10	187.75	194.10	1	48.7
R104	50	186.85	194.94	221.75	2	43.8	186.85	187.34	188.02	2	48.4	186.85	187.19	188.01	4	46.0	186.85	187.49	188.02	3	48.6	186.85	187.28	188.02	4	48.1
R108	50	188.49	213.28	252.48	1	38.2	188.49	190.32	197.61	5	47.6	188.49	206.04	221.72	1	49.8	188.49	189.30	195.60	2	53.9	188.49	195.99	220.46	2	58.9
R110	50	182.46	182.46	182.46	10	34.4	182.46	182.46	182.46	10	34.7	182.46	182.46	182.46	10	35.4	182.46	182.46	182.46	10	36.2	182.46	182.46	182.46	10	36.7
R207	50	231.55	240.12	248.50	0	43.8	231.01	234.78	235.92	0	61.6	231.01	239.58	245.66	0	62.4	231.01	235.42	241.28	0	52.9	230.94	234.61	235.92	1	60.8
R210	50	186.98	228.61	282.77	1	56.2	186.98	206.19	241.15	4	68.6	186.98	208.32	237.12	2	49.5	186.98	210.57	240.89	3	64.5	186.98	203.08	231.59	3	51.2
RC103	50	185.17	191.24	226.37	4	49.9	185.17	185.37	187.19	9	50.5	185.17	185.71	188.22	8	51.5	185.17	185.52	188.22	8	46.7	185.17	185.30	186.44	9	54.5
RC105	50	189.11	189.54	193.38	0	50.1	186.45	187.78	189.11	5	55.9	186.45	189.27	193.38	1	44.4	186.45	189.27	193.38	1	57.7	189.11	191.91	199.55	0	47.1
RC108	50	226.27	247.79	265.00	0	55.9	207.80	223.27	247.44	6	62.1	242.16	257.99	286.09	0	59.7	209.10	238.69	256.87	0	67.2	244.87	246.90	256.25	0	60.1
RC201	50	217.86	241.81	291.01	5	60.6	217.86	217.89	218.03	8	53.7	217.86	229.13	276.46	7	45.4	217.86	217.95	218.03	5	56.0	217.86	218.02	218.48	6	47.7
RC204	50	182.17	212.12	278.85	1	51.3	182.17	183.30	192.45	8	69.9	182.17	184.95	193.18	4	46.9	182.17	183.94	197.39	7	66.7	182.17	186.13	208.78	6	49.9
RC208	50	240.81	255.76	316.10	2	64.6	241.47	246.54	253.88	0	76.5	240.81	248.12	253.88	1	53.2	241.47	250.73	255.57	0	78.6	240.81	246.24	253.88	1	74.9

Table 14. Detailed results on medium scale VRPTW-based instances

		TS_C					$TS_{I\text{-}F\text{-}S}$					TS_{FLS}					TS_{ILS}					$TS_{F\text{-}I\text{-}P}$				
Instance	*De*	*Best*	*Avg*	*Worst*	*N*	*CPU*	*Best*	*Avg*	*Worst*	*N*	*CPU*	*Best*	*Avg*	*Worst*	*N*	*CPU*	*Best*	*Avg*	*Worst*	*N*	*CPU*	*Best*	*Avg*	*Worst*	*N*	*CPU*
C101	100	230.39	247.35	300.83	0	222.0	228.22	229.81	230.82	1	318.7	228.22	238.99	250.32	1	221.5	255.52	286.89	315.33	0	268.8	230.39	233.98	244.56	0	243.2
C104	100	228.13	231.42	251.88	6	208.1	228.13	228.13	228.13	10	212.8	228.13	232.64	251.87	7	247.9	228.13	234.19	257.76	1	198.8	228.13	228.79	231.42	8	184.0
C109	100	224.67	249.40	337.61	0	264.8	223.39	223.39	223.39	10	227.8	223.39	234.81	252.52	3	289.4	242.28	258.58	274.10	0	290.0	223.39	238.14	276.49	2	217.5
C201	100	218.16	252.04	276.90	0	242.2	215.25	218.45	223.11	1	345.7	220.26	248.40	284.76	0	217.2	251.60	279.55	313.82	0	287.5	218.45	230.24	247.11	0	269.6
C203	100	226.01	250.82	359.67	4	173.3	226.01	226.01	226.01	10	242.8	226.01	226.01	226.01	10	190.4	227.36	259.07	285.81	0	224.0	226.01	227.58	241.73	9	185.8
C208	100	214.55	239.92	271.49	0	254.8	199.33	203.20	205.02	0	429.6	196.00	220.83	234.17	1	270.9	205.02	242.36	295.94	0	370.3	204.67	211.58	231.43	0	247.9
R101	100	197.22	204.61	220.54	3	316.7	197.22	197.22	197.22	10	472.6	197.22	216.58	305.88	3	321.2	197.22	201.19	219.85	7	565.1	197.22	197.22	197.22	10	336.0
R104	100	200.73	201.91	210.00	5	302.4	200.73	200.81	201.52	9	360.0	200.73	206.94	232.62	4	303.3	203.43	214.43	222.91	0	268.5	200.73	201.37	206.91	7	346.2
R108	100	196.38	203.77	218.44	2	226.8	196.38	196.89	201.10	7	288.3	196.74	204.91	211.18	0	243.0	196.38	207.34	250.13	1	276.7	196.38	200.57	213.76	3	299.9
R110	100	228.08	229.10	238.19	8	206.6	228.08	228.08	228.08	10	230.9	228.08	229.57	235.46	1	215.2	228.20	237.90	263.19	3	260.3	228.08	230.76	237.85	4	248.6
R207	100	203.17	209.06	239.95	6	240.9	203.17	203.17	203.17	10	205.4	203.17	203.17	203.17	10	183.6	203.17	203.20	203.42	9	224.8	203.17	203.17	203.17	10	218.3
R210	100	208.84	223.65	242.50	2	216.0	208.84	211.56	215.63	6	205.3	208.84	230.13	246.23	1	235.2	215.63	231.21	245.42	0	210.4	208.84	219.45	245.42	2	212.6
RC103	100	245.46	265.78	294.05	0	300.5	245.07	246.82	252.69	1	370.2	245.46	249.36	258.82	1	221.4	254.71	283.53	307.69	1	349.1	249.63	261.38	298.34	1	377.8
RC105	100	195.64	226.76	273.15	0	344.8	193.09	194.28	196.57	1	506.6	197.59	220.01	256.87	0	357.5	205.11	225.82	282.00	0	341.5	198.86	214.77	225.80	0	357.9
RC108	100	241.51	246.59	266.30	5	210.2	241.51	241.51	241.51	10	233.3	241.51	244.88	252.78	5	225.8	247.05	252.66	268.21	0	260.4	241.51	245.09	257.06	4	235.7
RC201	100	220.59	243.81	263.16	0	287.5	227.10	231.95	236.46	0	417.1	214.22	243.01	276.22	1	301.2	249.88	317.02	367.18	0	240.6	243.61	289.27	347.56	0	263.2
RC204	100	228.22	242.73	258.90	3	248.5	228.22	228.22	228.22	10	257.6	228.22	231.14	250.01	8	328.5	228.22	238.86	258.20	3	282.2	228.22	228.22	228.22	10	277.8
RC208	100	228.25	229.19	236.45	8	187.3	228.25	228.25	228.25	10	189.8	228.25	228.25	228.25	10	231.4	228.25	243.52	304.80	3	256.0	228.25	228.25	228.25	10	196.1

Table 15. Detailed results on large scale VRPTW-based instances

		TS_C					$TS_{I\text{-}F\text{-}S}$					TS_{FLS}					TS_{ILS}					$TS_{F\text{-}I\text{-}P}$				
Instance	*De*	*Best*	*Avg*	*Worst*	*N*	*CPU*	*Best*	*Avg*	*Worst*	*N*	*CPU*	*Best*	*Avg*	*Worst*	*N*	*CPU*	*Best*	*Avg*	*Worst*	*N*	*CPU*	*Best*	*Avg*	*Worst*	*N*	*CPU*
C1_4_1	350	525.85	531.41	551.20	4	5137.9	525.85	530.11	549.86	4	6241.5	525.85	546.79	612.73	4	4107.8	553.93	608.91	674.62	0	4729.6	525.85	533.71	559.91	7	4785.8
C1_4_5	350	557.61	606.43	769.88	4	5088.3	557.61	558.75	565.94	7	5857.3	557.61	563.25	578.90	7	5427.0	599.11	636.21	672.47	0	4966.0	557.61	566.30	578.90	5	4362.2
C1_4_8	350	542.56	542.56	542.56	10	5208.8	542.56	542.56	542.56	10	5978.4	542.56	545.57	572.64	9	4401.4	555.11	651.26	805.94	0	5342.3	542.56	542.78	544.76	5	5442.0
C2_4_3	350	469.17	588.25	878.67	0	5258.5	469.11	469.95	474.20	7	5923.9	469.11	471.51	475.43	5	5553.4	538.10	604.56	654.02	0	5779.7	469.11	472.76	492.50	5	5663.2
C2_4_7	350	505.09	556.92	764.40	0	5891.3	507.76	523.36	542.87	0	6901.1	507.76	527.60	549.91	0	6381.2	533.38	601.88	659.84	0	5644.2	502.07	519.05	535.22	1	4602.0
C2_4_9	350	487.57	517.20	600.18	3	5211.4	487.57	497.46	514.91	1	5184.7	487.57	492.14	527.86	4	6258.6	500.31	547.53	610.77	0	5290.7	487.57	502.08	525.86	4	5644.1
R1_4_2	350	501.08	519.08	569.88	3	4860.2	505.27	510.63	518.88	0	6674.6	501.13	535.46	641.32	0	5075.8	557.63	596.26	665.74	0	4683.8	518.88	538.95	570.77	0	5001.5
R1_4_3	350	572.00	591.45	654.91	6	5431.9	572.00	572.00	572.00	10	5008.0	572.00	608.79	717.36	5	4209.3	584.42	607.63	615.46	0	5886.3	572.00	582.70	646.56	7	4576.2
R1_4_7	350	536.12	544.21	571.18	1	4744.6	536.12	536.60	539.21	2	5874.6	536.13	547.36	578.45	0	4706.5	596.59	626.75	692.43	0	4892.2	536.13	544.78	576.54	0	4817.5
R2_4_1	350	601.41	611.80	657.72	8	3831.1	601.41	601.41	601.41	10	5517.5	601.41	601.41	601.41	10	3763.4	614.31	692.81	763.59	0	5395.0	601.41	602.94	616.69	9	4091.8
R2_4_5	350	552.85	643.50	775.12	1	4860.2	552.85	592.26	661.72	1	6969.6	552.85	554.97	572.70	5	5725.5	598.55	641.73	716.12	0	5644.9	552.85	569.38	663.46	2	5106.8
R2_4_8	350	575.02	692.58	834.06	0	4063.5	558.88	562.11	574.81	7	4945.0	558.88	569.31	577.76	2	4153.0	677.84	717.39	840.33	0	6643.4	558.88	569.25	577.37	1	5195.4
RC1_4_3	350	570.71	581.48	609.91	4	5271.3	570.71	570.71	570.71	10	4973.0	570.71	588.14	635.32	3	4248.1	587.69	612.20	667.81	0	5621.6	570.71	571.28	576.05	7	5478.2
RC1_4_5	350	526.33	528.58	539.78	8	4468.2	526.33	526.33	526.33	10	6366.9	526.33	534.01	582.72	6	4772.2	584.14	609.52	636.41	0	5236.0	526.33	553.78	640.21	3	5928.5
RC1_4_8	350	558.88	560.64	574.13	8	3920.0	558.88	558.88	558.88	10	6230.7	558.88	561.59	574.13	5	4992.7	578.84	588.99	614.73	0	4503.7	558.88	573.08	623.80	2	4662.7
RC2_4_2	350	572.00	711.82	945.00	1	4879.2	572.00	572.00	572.00	10	6239.8	572.00	577.28	624.38	8	7163.2	604.23	650.93	679.80	0	5446.9	572.00	572.00	572.00	10	6120.7
RC2_4_5	350	522.34	575.12	783.77	4	4828.9	522.34	535.12	562.84	1	6204.6	522.34	526.04	540.46	7	5471.5	576.68	661.88	744.89	0	5863.6	522.34	522.94	528.36	9	5738.2
RC2_4_8	350	534.36	594.39	943.18	5	6189.5	534.36	537.98	543.06	2	6275.8	534.36	544.86	595.40	4	6104.7	636.13	706.85	752.86	0	6100.3	534.36	542.51	568.59	3	5603.9