Heuristic algorithms for a vehicle routing problem with simultaneous delivery and pickup and time windows in home health care

Ran Liu, Xiaolan Xie, Vincent Augusto, Carlos Rodriguez

Heuristic algorithms for a vehicle routing problem with simultaneous delivery and pickup and time windows in home health care

Ran LIU[a,b], Xiaolan XIE[a,b], Vincent AUGUSTO[a], Carlos RODRIGUEZ[a]

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

[b] Shanghai Jiao Tong University
Centre for Healthcare Engineering, Dept. Industrial Engr. & Logistics Management
800 Dongchuan Road, Min Hang District，200240 Shanghai，China

Correspondence: Prof. Xiaolan XIE, Centre for Health Engineering (CIS), Ecole Nationale Superieure des Mines de Saint-Etienne (ENSM.SE), 158 cours Fauriel, 42023 Saint-Etienne cedex 2 France. Tel. +33(0)477426695; Fax +33(0)477420249; xie@emse.fr

Submitted on 10/04/2012, revised 27/03/2013

Abstract: This paper addresses a vehicle scheduling problem encountered in home health care logistics. It concerns the delivery of drugs and medical devices from the home care company's pharmacy to patients' homes, delivery of special drugs from a hospital to patients, pickup of bio samples and unused drugs and medical devices from patients. The problem can be considered as a special vehicle routing problem with simultaneous delivery and pickup and time windows, with four types of demands: delivery from depot to patient, delivery from a hospital to patient, pickup from a patient to depot and pickup from a patient to a medical lab. Each patient is visited by one vehicle and each vehicle visits each node at most once. Patients are associated with time windows and vehicles with capacity. Two mixed-integer programming models are proposed. We then propose a Genetic Algorithm (GA) and a Tabu Search (TS) method. The GA is based on a permutation chromosome, a split procedure and local search. The TS is based on route assignment attributes of patients, an augmented cost function, route re-optimization, and attribute-based aspiration levels. These approaches are tested on test instances derived from existing VRPTW benchmarks.

Keywords: Home health care logistics, vehicle routing, pickup and delivery, time windows, metaheuristics,

1 Introduction

This paper considers a special vehicle routing problem with simultaneous delivery and pickup and time windows in home health care. Home hospitalization organizations have been created for patients requiring long and regular health cares in order to provide quality health service at their home while reducing the bed requirements at hospitals. Home hospitalization initially focused on nursing cares and has been extended to complex and technical cares such as chronic cares, rehabilitation, end-of-life palliative cares, and home chemotherapy. Home health care services are provided in France by Home Health Care (HHC) companies. Each day, a HHC company has various logistic activities including delivering drugs and medical devices from its pharmacy (also called depot in this paper) to patients at their home. It also takes some special drugs, such as chemotherapy drugs and blood products, from hospitals to patients. On the other hand, the HHC also needs to pick up materials from patients and deliver to different locations. Blood samples of the patients are collected and delivered to a medical lab. Medical wastes, unused drugs and medical devices are collected and brought back to the HHC or the depot. As HHC companies are usually small but serve rather large number of patients with dispersed locations, it is crucial to carefully design the routes of the HHC vehicles in order to reduce its operating cost while improving the service quality to patients.

Since an HHC patient may be a delivery and a pickup client simultaneously and have both pickup and delivery demands, the design of HHC vehicle routes is related to the *vehicle routing problem with simultaneous pickup and delivery and time windows* (VRPSDPTW) introduced by (Hokey, 1989). The VRPSDPTW is a hard and challenging problem in the field of vehicle routing problem (VRP). It considers clients that require simultaneous pickup and delivery service. Some common constraints must be satisfied in both HHC's vehicle scheduling problem and the VRPSDPTW. For example, each client must be visited and served in a given time window; the load on a vehicle must always be below the vehicle capacity. However, the problem faced by the HHC company is more complex than the classical VRPSDPTW. The first reason is the complexity of its logistic operations with different types of pickup and delivery demands of patients. According to the origins and destinations, both pickup and delivery demands can be divided into two subclasses. The pickup demands include: (i) picking up the material from patients' homes and deliver to a lab, e.g., biological samples; (ii) picking up some materials from the patients' homes and bring back to the depot, e.g., medical waste. Similarly, there are two subclasses of delivery demands required by the patients: (i) delivering the products from the company's depot to patients; (ii) delivering some materials from a hospital to patients' homes, e.g., special drugs for cancer treatment. In the classical VRPSDPTW all delivery goods are loaded at depot and all pickup goods have to be transported to depot. In our HHC vehicle scheduling problem besides the depot, goods can be transported from a hospital to patients and from the patients to a lab. Clearly, the composition of vehicles' loads in our case is more complex than VRPSDPTW. Furthermore, different from the classical VRPSDPTW, each route of our problem must satisfy some *precedence* *constraints*, e.g., for a patient who needs drugs provided by the hospital, the vehicle visiting the patient has to visit the hospital first. Such special constraints are similar to the pairing and precedence constraints in classical *pickup and delivery problem* (PDP), in which each customer request is defined by an origin location and a destination, the origin must visited before the destination by the same vehicle. However, the PDP is less complicated than our problem, since the origins as well as the destinations of transportation requests in the PDP are locations other than the depot, and a customer in the PDP only has either pickup or delivery request. Thus, in this paper the HHC vehicle scheduling problem is rather a special VRPSDPTW variant which has never been studied before. Since both the VRPSDPTW and PDP are NP-hard problems, our problem is more complex than these problems and is also NP-hard. To the best of our knowledge, we have not found any existing work dealing this special simultaneous pickup and delivery problem in HHC industry.

In this paper, we first perform a literature review, propose two mathematical formulations of our problem, and then develop two heuristic algorithms for this special vehicle scheduling problem. The rest of this paper is organized as follows. Section 2 introduces the relevant literature. Our problem is formally defined and two mathematical models are given in Section 3. Section 4 proposes a Genetic algorithm (GA) for our problem. Section 5 proposes a Tabu Search (TS) algorithm for solving the problem. Computational experiments are described in Section 6. Section 7 concludes the paper.

2 Literature review

As stated before, two main bodies of vehicle routing literature are relevant to our problem. The first is the *vehicle routing problem with simultaneous pickup and delivery and time windows*, in which goods are transported by a fleet of vehicles between the depot and customers within their time windows. The second one is the *pickup and delivery problem with time windows* (PDPTW) problem, in which goods are transported between *n* pickup and *n* delivery locations, and the vehicle visiting each location must be within an associated time window. We survey the literature in two parts.

The VRPSDPTW is an extension of the VRPSDP, and has been much less studied than the VRPSDP. The VRPSDP can be seen an extension of the *vehicle routing problems with backhauls* (VRPB). In the VRPB, the set of customers are divided in two subsets consisting of *linehaul* and *backhaul* costumers, where a linehaul customer requires a given quantity of product to be delivered from the depot, and a backhaul customer requires a given quantity of product to be picked up to the depot. In the VRPB, it is assumed that the vehicles only pick goods up (serve backhaul customers) after they have finished delivering their entire load (serve linehaul customers) (P. Toth and Vigo, 1997a) (Goetschalckx and Jacobs-Blecha, 1989). One reason for this assumption is the difficulty to re-arrange delivery and pickup goods on the vehicles. The objective of the VRPB is to design a set of minimum cost routes so that on each route neither the total load of linehaul customers nor that of backhaul customers exceed the vehicle capacity. The VRPB is a NP-hard problem in strong sense and a number of algorithms are proposed for this problem. Exact methods for the VRPB are proposed by (Yano, et al., 1987) (P. Toth and Vigo, 1997a) (Mingozzi, et al., 1999). Heuristics have been developed by (Goetschalckx and Jacobs-Blecha, 1989) (Paolo Toth and Vigo, 1999) (Osman and Wassan, 2002) (Tavakkoli-Moghaddam, et al., 2006) (Y Gajpal and PL Abad, 2009). If the linehaul and backhaul customers can be freely mixed within a route, the VRPB is transformed to the *vehicle routing problem with mixed backhauls* (VRPMB). Clearly, the vehicle capacity check in the VRPMB is more complicated than the VRPB. Exact solution methods for the VRPMB have only been developed for the single vehicle case (Eilam Tzoreff, et al., 2002) (Süral and Bookbinder, 2003) (Baldacci, et al., 2003). Heuristics for the VRPMB are given by (Nagy and Salhi, 2005) (Salhi and Nagy, 1999) (Wade and Salhi, 2004) (Reimann and Ulrich, 2006). Based on the VRPMB, if we allow customers to have both pickup quantity and delivery quantity, then, there exist two special problems: the *vehicle routing problem with divisible delivery and pickup* (VRPDDP), and *the vehicle routing problem with simultaneous delivery and pickup* (VRPSDP). The difference between the VRPDDP and VRPSDP is the number of times a customer is visited. In the VRPDDP customers do not have to be visited exactly once, i.e., a customer can be visited twice, once for pickup and once for delivery service. The VRPSDP requires that each customer is visited only once by a vehicle. The VRPDDP instances can be transformed to VRPMB by modeling every customer’s pickup and delivery service as two separate customers. One exact method for the VRPSDP was designed by (Dell’Amico, et al., 2006). (Dethloff, 2002) proposes an extension of the cheapest insertion heuristic to the VRPSDP. Several tabu search algorithms for VRPSDP were proposed in (Alfredo Tang Montané and Galvão, 2006) (Chen and Wu, 2005) (Bianchessi and Righini, 2007) (Crispim and Brandão, 2005). Recently, (Ai and Kachitvichyanukul, 2009) (Y. Gajpal and P. Abad, 2009) (Subramanian, et al., 2010) have proposed several metaheuristics to solve VRPSDP.

Contrary to the VRPB, VRPMB, and VRPSDP, only a fewer researchers consider the time window constraints in these problems, especially for the VRPSDP. For example, an exact algorithm was designed for the *VRPB with time windows* in (Gélinas, et al., 1995). (Duhamel, et al., 1997) (REIMANN, et al., 2002) (Thangiah, et al., 1996) (Zhong and Cole, 2005) proposed heuristics for this problem. For the *VRPMB with time windows,* (Hasama, et al., 1998) (Kontoravdis and Bard, 1995) (Zhong and Cole, 2005) designed heuristics, with the primary objective of minimizing the number of the vehicles and the second objective of minimizing traveling distances. For the most complex one, VRPSDPTW, only (Angelelli and Mansini, 2002) proposed an exact method and (Mingyong and Erbao, 2010) and (Wang and Chen, 2012) proposed genetic algorithms.

Compared with VRPSDPTW, there is an abundant body of research on the second related problem PDPTW. The PDPTW originates from the basic PDP (M. W. P. Savelsbergh and Sol, 1995). In the PDP, a customer order consists of two parts: a pickup at one location and a delivery at another location. The PDP has been intensively studied in the past three decades. For survey on the PDP, the reader is referred to (M. W. P. Savelsbergh and Sol, 1995), (Berbeglia, et al., 2007), and (S. Parragh, et al., 2008). For the PDPTW, several exact approaches have been designed. (Dumas, et al., 1991) (M. Savelsbergh and Sol, 1998) (Xu, et al., 2003) (Sigurd and Pisinger, 2004) used branch and price schemes for the PDPTW. (J.-F. Cordeau, 2006) (Ropke, et al., 2007) developed branch and cut approach for the PDPTW. (Ropke and Cordeau, 2009) introduced a new branch and cut and price algorithm for the PDPTW. Meanwhile, many heuristics have proposed for the PDPTW. (Jaw, et al., 1986), (Madsen, et al., 1995), (Diana and Dessouky, 2004), (Lu and Dessouky, 2006) presented various insertion-based heuristics for solving the PDPTW. (P. Toth and Vigo, 1997b) (Nanry and Wesley Barnes, 2000) (J.-F. Cordeau and Laporte, 2003) solved the PDPTW by means of tabu search heuristics. (Li and Lim, 2001), (Pankratz, 2005), (Ropke and Pisinger, 2006) and (S. N. Parragh, et al., 2010) designed simulated annealing, genetic algorithm, adaptive large neighborhood search heuristic, and variable neighborhood search heuristic for solving the PDPTW. Extensive review of the PDPTW literature is out of the scope of in this paper and interested readers are referred to the surveys of (Berbeglia, et al., 2007) and (S. Parragh, et al., 2008).

Although a large number of literatures have studied the VRPSDPTW and PDPTW, to our best knowledge, no existing literature consider the problem, which contains following two vehicle service strategies simultaneously: (i) the transportation of goods from the depot to linehaul customers and from backhaul customers back to the depot; (ii) goods are transported between pickup and delivery locations. Our HHC vehicle scheduling problem includes both of these two service strategies and can be seen as a special VRPSDPTW variant.

3 Mathematical Formulation

This paper addresses the daily scheduling problem of vehicles of a home health care company for delivery of drugs and medical devices and for pickup of biological samples and medical wastes or unused drugs. This section provides a formal description of the problem and then presents two equivalent mixed integer programming formulations of the problem that will serve to assess the efficiency of the heuristic methods of this paper.

The problem can be defined as follows. Let $G = (V, A)$ be a directed graph with a set $V = \{0, 1, ..., n, n+1\} \cup \{h, l\}$ of nodes and a set $A = \{(i, j): i, j \in V, i \neq j\}$ of arcs. Nodes 0 and $n+1$ represent the origin and destination depots which are in practice the pharmacy of the home health care company. Each vehicle starts at node 0 and ends at node $n+1$. Nodes $N = \{1,..., n\}$ correspond to patients' homes. Node h and l represent the locations of a hospital and a medical lab.

Each patient $i \in N$ has four types of delivery and pickup requirements: d_{i1}, d_{i2}, p_{i1} and p_{i2}, where d_{i1} represents the amount of materials (drugs/medical devices) to deliver from the depot 0 to patient i, d_{i2} the amount of materials (special drugs) to deliver from the hospital to patient i, p_{i1} the amount of materials to pick up from patient i and bring back to the depot $n+1$, and p_{i2} the amount of materials (biological samples) to pick up from patient i and bring to the medical lab *h*. Each type of requirements is called a *demand*. Different materials are assumed to be compatible and can be loaded in the same vehicle. $D_1 \subseteq N$ denotes the set of patients needing type 1 delivery service, i.e. patients *i* with $d_{i1} > 0$. Similarly, D_2, P_1, P_2 denote sets of patients needing type 2 delivery, type 1 and type 2 pickup services. A patient may require different types of demands. For example, for a patient $i \in D_2 \cap P_2$, the company has to pick up the quantity p_{i2} from node *i* and deliver to the lab and deliver the quantity d_{i2} from the hospital to this node. For notation convenience, we set zero-demands for nodes 0, $n+1$, *h, l.*

A time window $[a_i, b_i]$ is associated with each node $i \in V$, where a_i and b_i represent the earliest and latest time. A vehicle is allowed to arrive before a_i and wait until the patient becomes available, but arrivals after b_i are prohibited. The depot node also has a time window, representing the earliest and latest times when the vehicles may leave from and return to the depot. Each arc $(i, j) \in A$ is associated with a routing cost c_{ij} and a travel time t_{ij}. The service time for a patient *i* is assumed to be included in the travel time t_{ij}.

A fleet *K* of identical vehicles, initially located at the depot, is available to serve the patients. Each vehicle has a capacity of *Q*.

The problem consists in determining a set of at most *K* routes of minimal overall cost in order to serve all delivery and pickup demands of all patients under the obvious time window and vehicle capacity constraints and the following assumptions.

Assumption A. Each route starts and ends the depot and visits each location at most once;

Assumption B. Each patient is visited by exactly one vehicle for all its demands;

Assumption C. Each route makes a hospital visit before visits to its D_2-patients;

Assumption D. Each route makes a lab visit after all visits to its P_2-patients.

A typical route is as follows. The vehicle starts the depot with all materials for its D_1-patients, visits some patients for D_1-delivery and any pickup, visits the hospital to load all materials for its D_2-patients, visits other patients, then goes to the lab to deliver materials of all P_2-patients, visits other patients before returning to the depot with all materials of its P_1-patients.

In the following, we propose two three-index MIP formulations. The first mathematical formulation termed MIP1 is derived from the model of (Dell'Amico, et al., 2006) and (Ropke and Cordeau, 2009). Four types of decision variables are used.

x_{ij}^k	binary variable equal to 1 if vehicle *k* travels directly from node *i* to node *j*;
B_i^k	time at which vehicle *k* begins to serve at node *i*;
y_{ij}^k	quantity of P_1- and P_2-pickup carried along arc (i, j) by vehicle *k*;
w_{ij}^k	quantity of D_1- and D_2-delivery carried along arc (i, j) by vehicle *k*.

MIP1:
$$Min \sum_{i\in V}\sum_{j\in V}\sum_{k\in K} c_{ij} x_{ij}^k \qquad (1)$$

Subject to:

$$\sum_{k\in K}\sum_{j\in V} x_{ji}^k = 1 \qquad \forall i \in N \tag{2}$$

$$\sum_{j\in V} x_{ij}^k \le \sum_{j\in V} x_{jl}^k \qquad \forall i \in P_2, k \in K \tag{3}$$

$$\sum_{j\in V} x_{ji}^k \le \sum_{j\in V} x_{hj}^k \qquad \forall i \in D_2, k \in K \tag{4}$$

$$\sum_{i\in V} x_{0i}^k \le 1 \qquad \forall k \in K \tag{5}$$

$$\sum_{i\in V} x_{i,n+1}^k \le 1 \qquad \forall k \in K \tag{6}$$

$$\sum_{j\in V} x_{ji}^k = \sum_{j\in V} x_{ij}^k \qquad \forall i \in N \cup \{l,h\}, k \in K \tag{7}$$

$$\sum_{j\in V} x_{ji}^k \le 1 \qquad \forall i \in \{l,h\}, k \in K \tag{8}$$

$$B_j^k \ge (B_i^k + t_{ij}) x_{ij}^k \qquad \forall i \in V, j \in V, i \ne j, k \in K \tag{9}$$

$$\sum_{i\in V}\sum_{k\in K} y_{ji}^k - \sum_{i\in V}\sum_{k\in K} y_{ij}^k = p_{j1} + p_{j2} \qquad \forall j \in N \cup \{h\} \tag{10}$$

$$\sum_{i\in V}\sum_{k\in K} w_{ij}^k - \sum_{i\in V}\sum_{k\in K} w_{ji}^k = d_{j1} + d_{j2} \qquad \forall j \in N \cup \{l\} \tag{11}$$

$$\sum_{i\in V} w_{hi}^k - \sum_{i\in V} w_{ih}^k = \sum_{i\in V}\sum_{j\in D_2} x_{ij}^k d_{i2} \qquad \forall k \in K \tag{12}$$

$$\sum_{i\in V} y_{il}^k - \sum_{i\in V} y_{li}^k = \sum_{i\in V}\sum_{j\in P_2} x_{ij}^k p_{i2} \qquad \forall k \in K \tag{13}$$

$$B_l^k \ge (B_i^k + t_{il}) \sum_{j\in V} x_{ij}^k \qquad \forall i \in P_2, k \in K \tag{14}$$

$$B_i^k \ge (B_h^k + t_{hi}) \sum_{j\in V} x_{ij}^k \qquad \forall i \in D_2, k \in K \tag{15}$$

$$a_i \le B_i^k \le b_i \qquad \forall i \in V, k \in K \tag{16}$$

$$y_{ij}^k + w_{ij}^k \le Q x_{ij}^k \qquad \forall i \in V, j \in V, k \in K \tag{17}$$

$$x_{ij}^k \in \{0,1\} \qquad \forall i, j \in V, i \ne j, k \in K \tag{18}$$

The objective function (1) minimizes the total routing cost. Constraints (2) ensure that all the demands are satisfied. Constraints (3) guarantee that a vehicle visiting a P_2-patient also visits the lab. Constraints (4) are similar but for hospital visit. Constraints (5) and (6) force the route of each vehicle to start and end at the depot. Constraints (7) ensure the flow balance of the vehicles, i.e., if a vehicle visits a node it must leave this node. Constraints (8) indicate that each vehicle can only visit the hospital and the lab once. Constraints (9) impose the consistency of the visiting times. Constraints (10) and (11) are flow equations for pickup and delivery demands. Constraints (12) impose that all delivery demands of D_2 patients are loaded at the hospital. Constraints (13) impose the unloading of all pickup demands of P_2 patients at the lab. Constraints (14) ensure that each P_2 patient is visited before a lab visit; constraints (15) ensure the hospital visit before any visit to a D_2 patient. Finally, constraints (16) and (17) impose the time window and vehicle capacity constraints. This MIP1 model is nonlinear because of constraints(9), (14) and (15) that can be linearized as follows:

$$B_j^k \geq B_i^k + t_{ij} - M(1 - x_{ij}^k) \qquad \forall i \in V, j \in V \setminus 0, k \in K \tag{19}$$

$$B_l^k \geq B_i^k + t_{il} - M(1 - \sum_{j \in V} x_{ij}^k) \qquad \forall i \in P_2, k \in K \tag{20}$$

$$B_i^k \geq B_h^k + t_{hi} - M(1 - \sum_{j \in V} x_{ij}^k) \qquad \forall i \in D_2, k \in K \tag{21}$$

The second mathematical formulation termed MIP2 uses decision variables x_{ij}^k and B_i^k as in MIP1 plus the following new decision variables:

z_{ij}^k : quantity of P_j-pickup with $j \in \{1,2\}$ carried by vehicle k when leaving node i;

v_{ij}^k : quantity of D_j-delivery with $j \in \{1,2\}$ carried by vehicle k when leaving node i.

MIP2: $Min \sum_{i \in V} \sum_{j \in V} \sum_{k \in K} c_{ij} x_{ij}^k$

subject to constraints (2)-(9), (14)-(16), (18) and

$$z_{j1}^k \geq (z_{i1}^k + p_{j1}) x_{ij}^k \qquad \forall i \in V, j \in N \cup \{l, h\}, i \neq j, k \in K \tag{22}$$

$$z_{j2}^k \geq (z_{i2}^k + p_{j2}) x_{ij}^k \qquad \forall i \in V \setminus \{l\}, j \in N \cup \{l, h\}, i \neq j, k \in K \tag{23}$$

$$v_{i1}^k \geq (v_{j1}^k + d_{j1}) x_{ij}^k \qquad \forall i \in V, j \in N \cup \{l, h\}, i \neq j, k \in K \tag{24}$$

$$v_{i2}^k \geq (v_{j2}^k + d_{j2}) x_{ij}^k \qquad \forall j \in V \setminus \{h\}, i \in N \cup \{l, h\}, i \neq j, k \in K \tag{25}$$

$$z_{i1}^k + z_{i2}^k + v_{i1}^k + v_{i2}^k \leq Q \qquad \forall i \in V, k \in K \tag{26}$$

Constraints (22) determine the vehicle loading for P_1-pickup. Constraints (23) track the vehicle loading for P_2-pickup and ensure that it becomes null after the lab visit at the optimum. Constraints (24) and (25) determine the vehicle loading for D_1 and D_2 delivery. Constraints (26) are vehicle capacity constraints. Again nonlinear constraints (22)-(25) can easily be linearized by standard reformulation techniques.

As explained in Sections 1 and 2, our problem is related to highly complex vehicle routing problems including VRPSDPTW and PDPTW problems. It can be easily proved that our problem is strongly NP-hard as it covers classical VRP problems as special case. It is expected that the above MIP formulation can only be used to solve small-size problems. This is evidence by numerical results of Section 6 in which we try to solve different test instances with the Cplex 12.3 solver. Even for instances of very small size with 30 patients and 40 demands, Cplex is not able solve the problem optimally. For some of the small size instances, it even cannot get a feasible solution in reasonable time (about 48 hours). For these reasons, we propose in the following a genetic algorithm and a tabu search method to address problems of large size.

4 A genetic algorithm

The genetic algorithm proposed in this paper combines the following features: a permutation chromosome, exact fitness computation by splitting, improvement by local search, diversity of the population and mutli-start with partially replaced population. These ingredients were shown powerful in (Prins, 2004) in designing efficient GA for vehicle-routing like problems.

In this paper, the chromosome is a permutation of all patients that give order of visits in different routes. An exact split algorithm will be presented to split the permutation into sub-strings of patients to be visited by a vehicle and to insert hospitals and labs.

Algorithm 1: Outline of the GA

```
1:  Generate an initial population $\Pi$ of chromosomes
Main GA exploration phase
2:  Select two parents $P_1$ and $P_2$ by binary tournament;
3:  Crossover ($P_1$, $P_2$) by OX operator;
4:  Evaluate the two resulting children by Splitting;
5:  Repeat 2-4 if no child is feasible. Otherwise, select randomly a feasible child $C$;
6:  Improve $C$ by Local Search, with probability $p_m$;
7:  Insert $C$ in $\Pi$ to replace a randomly selected individual among the half worst of $\Pi$, if
     $C$ is not the current worst and C has a distinct fitness value than those in $\Pi$;
8:  Repeat 2-7 for $N_1$ iterations or till $N_2$ iterations without improving the current best;
End of the main GA phase
9:  Restart the main GA phase 2-8 with a partially replaced population, for $N_3$ phases or
     till $N_4$ phases without improving the best solution.
```

The overall structure of the GA is illustrated in Algorithm 1. It starts with the generation of an initial population with insertion of good heuristic solutions to be presented. The central part of our GA is an incremental GA exploration phase in which only one chromosome is replaced at each iteration. It starts with the selection of two parents by binary tournament. The OX operator that was proved appropriate for VRP problems in (Prins, 2004) is then applied to generate two child chromosomes. These child chromosomes are evaluated by the exact split algorithm. The selection and crossover operations are repeated till obtaining a feasible child chromosome, i.e. a chromosome for which a feasible split solution exists. With some probability, this feasible child chromosome is further improved by Local Search to be presented. The new child chromosome is inserted in the current population under two conditions: (i) it is better than the current worst and (ii) it does not have identical fitness as an existing chromosome.

In our GA implementation, the main GA phase is performed for N_1=3000 iterations or till N_2=1500 iterations without improving the current best. The main GA phase is repeated for

N_3=10 phases or till N_4=5 phases without improving the current best. To restart the GA phase, we keep the ψ best chromosomes with ψ =3 and replace the others with randomly generated chromosomes. Through some preliminary experiments, the proposed GA gives better solution when increasing the local search probability p_m and the size of the population $|\Pi|$, especially when p_m is smaller than 0.8 and $|\Pi|$ is less than 35. The negative side is the increasing running time of GA. To balance the accuracy and the speed of GA, the local search is applied with a probability p_m = 0.8, and the size of the population is kept fixed at $|\Pi|$ = 30 chromosomes.

It has been proven that perverting the diversity of GA population can diminish the risk of premature convergence (Sörensen and Sevaux, 2006). A simple and stricter rule is imposed in this paper to keep the diversity of the population, i.e., the fitness of any two feasible chromosomes must be different. For this reason, a child chromosome C is inserted during each main GA phase only if it has a different fitness than existing individuals. Diversity is also checked in the generation of the initial population and the partially replaced population for restart.

The remaining of this Section is devoted the detailed presentation of the fitness evaluation, generation of initial solutions, and local search.

4.1 The chromosome and fitness evaluation

In our GA, a chromosome C is a permutation (s_1, $s_2 \dots s_n$) of all patients (1, 2, …, n) and the fitness is the optimal criterion value of our problem such that each route visits a sub-string of patients of C and in the order of C with of course necessary visits to the hospital and lab. For example, the solution of a chromosome (3, 5, 1, 2, 4) could be (0, 3, 5, h, 1, n+1) and (0, 2, 4, l, n+1).

Hereafter we propose a split procedure for fitness evaluation of our problem. It is a shortest path approach. It first builds an auxiliary graph H which contains nodes (s_0, s_1, $s_2 \dots$, s_n) with $s_0 = 0$ and in which each arc corresponds to a feasible route. There is an arc (s_i, s_j) from node s_i to s_j with $i < j$ if there exists a feasible route visiting all patients (s_{i+1}, …, s_j) in the given order. The length of the arc is the minimal total routing cost of such a route. The fitness of the chromosome is the shortest path from s_0 to s_n with at most K arcs.

In this approach, when considering an arc (s_i, s_j), all possible and necessary insertions of the hospital h and the lab l in the route (0, s_{i+1},…, s_j, n+1) are considered. For each insertion of (h, l), the earliest visiting time of each node is determined, the corresponding time window constraint checked and the insertion is abandoned if the time window constraint is violated. The length l_{ij} of the arc (s_i, s_j) is the smallest earliest visiting time of node n+1 among all feasible routes.

Once the auxiliary graph H is built, the fitness is determined by dynamic programming by determining iteratively the shortest path SP_{ik} from s_0 to s_i with at most k arcs. Clearly, the fitness is SP_{nK} determined by the following recursion:

$$\begin{cases} SP_{jk+1} = \underset{i<j}{MIN}\, SP_{ik} + l_{ij}, \forall k = 1,\ldots,K-1 \\ SP_{00} = 0 \end{cases}$$

Figure 1 is an illustrative example. The first part of Figure 1 shows a small instance with a depot(*D*), a hospital(*h*), a lab(*l*), three patients (*a*, *b*, *c*) and 2 identical vehicles with capacity *Q*=10. The chromosome is (*a*, *b*, *c*). The distances between any two nodes are given beside the arc, and the demand of each patient is shown in brackets. There are no time window constraints. In the second part of Figure 1, each arc in the auxiliary graph represents a possible vehicle route and its traveling distance. The route visiting patients *a* and *b*, i.e., arc (*D*, *b*), has two possible positions for visiting the hospital and the one leading to shortest distance is chosen to represent this arc. Based on the second part, is the shortest path with no more than 2 arcs is (D−*a*−*c*) and has a cost of 52. The corresponding solution consisting of two vehicle routes is given in the third part of Figure 1.

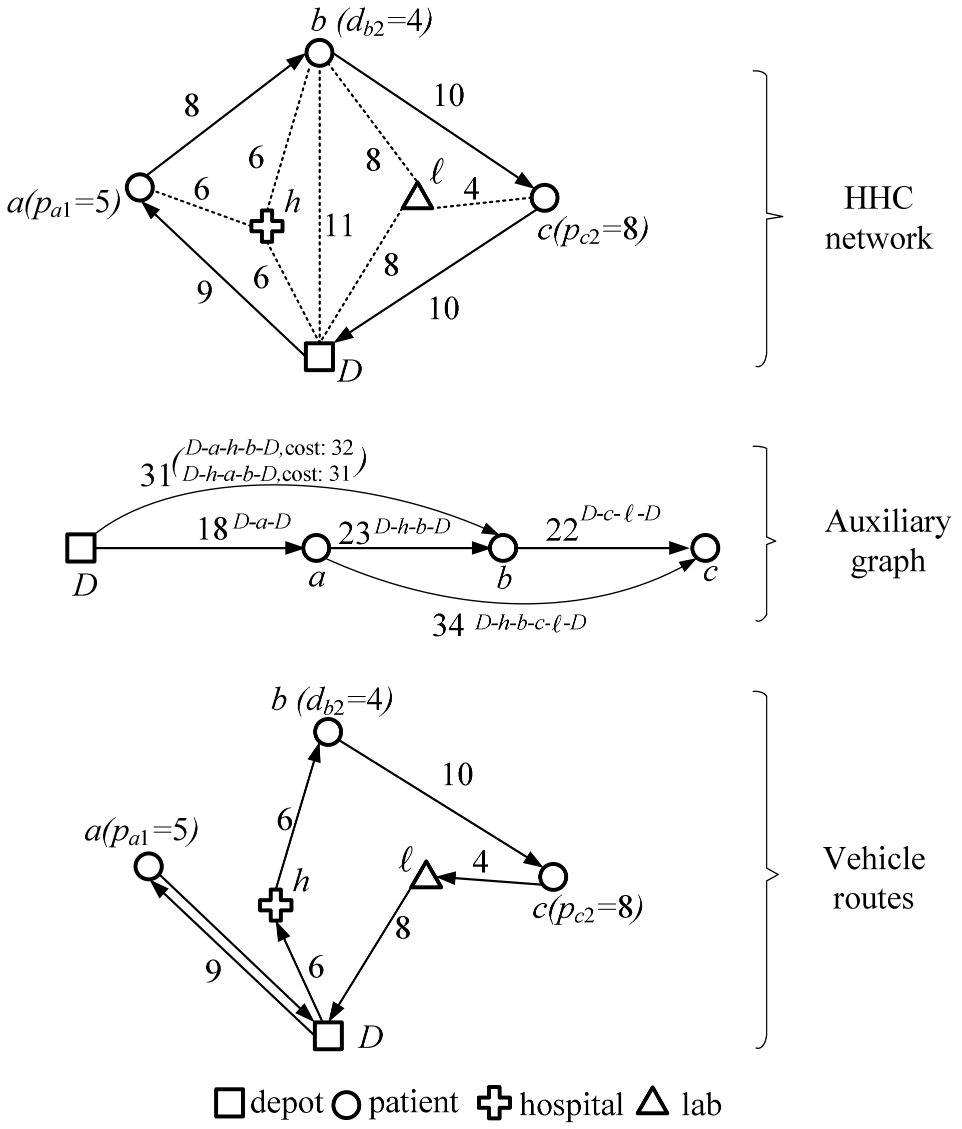

Figure1. Split procedure for the chromosome (a, b, c)

4.2 Constructing the initial population

The initial population is a combination of heuristic solutions and randomly generated chromosomes. It uses four simple constructive heuristics including three saving-based heuristics and a *Nearest Neighbor* (*NN*) heuristic. The heuristics make use of randomly set parameters but details will be given later.

The initial population is generated in two phases. Phase 1 starts with a population of the four heuristic solutions with randomly set parameters and randomly generated chromosomes. Phase 1 repeats till a feasible chromosome is obtained. Phase 2 tries to ensure the diversity of the population. It replaces two types of individuals with new random chromosomes: feasible chromosomes having the same fitness as another one and infeasible chromosomes. Phase 2 is repeated for $N_5 = 100$ times or till the diversity is ensured.

When restarting the GA exploration phase, the current population is partially replaced by keeping the ψ best chromosomes, replacing the remaining ones with random chromosomes, and applying phase 2 to ensure the diversity.

We now give details of the four construction heuristics. The first simplest saving method, called $saving_1$, is as follows.

Step1 assigns each patient to a separate route. Each patient of P_1 and D_1 is connected with the depot, forming a route beginning and ending at depot. Each P_2 patient j is connected with the lab and the depot, forming a route (depot−j−l−depot). Similarly, each patient of D_2 is connected with the hospital and the depot;

Step2 merges two routes associated with the maximal saving value, which is calculated as in the classical saving method by checking the time windows and vehicle capacity constraints. In the classical saving, when two routes (depot−…−i−depot) and (depot−j−…−depot) can be feasibly merged into a single route (depot−…−i−j−…depot), the saving value is $c_{i0}+c_{0j}-\lambda_1 c_{ij}$ where λ_1 is a parameter randomly sampled in [0.8, 1]. In our problem, the saving value is equal to the total distance of arcs to delete in the merged route minus the distances of arcs to add in the new giant route. If both two routes contain the hospital (lab), only the first hospital (the second lab) is kept in the new giant route. For example, the left part of Figure2 show two routes (depot−h−…−i−depot) and (depot−h−j−…−depot) where i and j are D_2 patients and the others are D_1 or P_1 patients. When merging two routes into one giant route (shown in the right part), three arcs are deleted (represented by dotted line), and one arc (i, j) is inserted into the new route. Thus, the saving value is $\Delta_1=c_{i0}+c_{hj}+c_{0h}-\lambda_1 c_{ij}$.

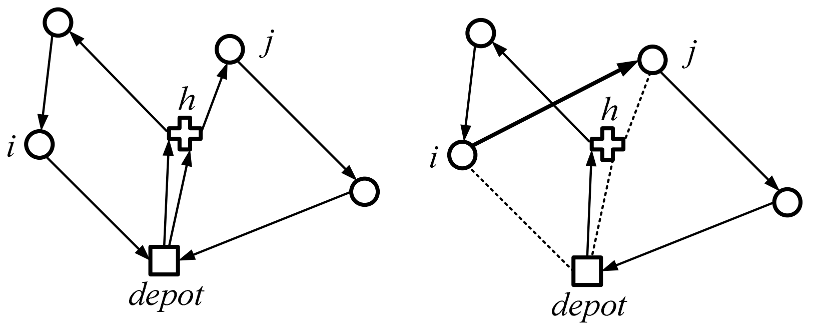

Figure2. The example of saving methods

Step3: repeat merging routes until no improvement is possible. The final chromosome is obtained by concatenation of the resulting routes in the linguistic order of the first patient in each route.

Two other saving heuristic *saving*$_2$ and *saving*$_3$ are also implemented with different one-step look-ahead criterion to improve saving1. *Saving*$_2$ combines the immediate saving value Δ_1 of merging two routes and the maximum positive saving value $\Delta_{1\text{-max}}$ of this giant route with other remaining routes in the next step. *Saving*$_2$ merges two routes with the largest $\Delta_2=\Delta_1+\lambda_2\Delta_{1-\max}$ among all merging with positive Δ_1 where λ_2 is the parameter, randomly sampled in [0.1, 0.3]. *Saving*$_3$ is similar with Δ_2 replacing by $\Delta_3=\Delta_1+\lambda_3\Delta_{1\text{-sum}}$ where λ_3 is a third parameter, randomly sampled in [0.01, 0.03] $\Delta_{1\text{-sum}}$ is the sum of top-10 positive savings of merging the giant route and remaining routes in the next step.

In the last heuristic *NN*, we only consider the depot and patients and neglect the lab and hospital. First, *NN* links the depot to its closest patient, i.e. the patient *i* with the smallest routing cost from 0 to *i*. Then, the *NN* adds a directed path connecting the last added node to the closest unvisited patient. The procedure repeats until all the patients are included in the path. The resulting sequence of patients is the chromosome of the *NN* heuristic.

4.3 Improving a feasible chromosome by local search (LS)

This subsection starts with the split solution of a feasible chromosome and then improves it by several local moves including 1-1exchange, 1-0 relocation, 2-Opt exchange, and 2-Opt* exchange. The *first-accept* strategy is used, i.e., once a new better neighboring solution is identified, it replaces the current solution. For each solution, it first tries 1-1exchange and accept the first improving 1-1exchange. If no improvement is possible with 1-1exchange, it then tries the other local moves. The process repeats until no improvement is possible at all.

For 1-1exchange and 1-0 relocation, both *intra-route* and *inter-route* movements are tried. The 1-1exchange tries to exchange positions of any two patients. The 1-0 relocation is the operator for one patient and transfers a patient from its position in one route to another position either the same or a different route. The 2-Opt exchange is only executed on a single route and it tries to improve the route by replacing two of its edges by two other edges. The 2-Opt* exchange is an inter-route method: two edges are selected from two different routes, respectively, and the end portions of two edges are exchanged, so as to generate two new routes.

Note that the inter-route 1-0 relocation and 2-Opt* may combine two routes into one and reduce the usage of the vehicles in the solution. In such case, the empty route is removed from the solution. Compared with classical VRPTW, in our work additional precedence constraints also must be checked during the LS procedure. During the search procedure, once a neighboring solution is identified, we checked whether it can satisfy the vehicle capacity, time windows, and precedence constraints. If not, it is aborted directly.

When the LS procedure stops, the resulting routes are concatenated in linguistic order to give the improved chromosome. The split procedure is applied to this new chromosome for evaluation of its fitness.

5 A tabu search algorithm

Tabu search (TS) is also one of the most powerful and competitive heuristics in the fields of VRP, VRPTW and other VRP variants (Gendreau, et al., 1994) (J. Cordeau, et al., 1997) (Hertz, et al., 2000) (Côté and Potvin, 2009) (Zachariadis, et al., 2009). Essentially, TS iteratively explores the solution space by moving from the current solution to another solution in its neighborhood. Since the current solution may deteriorate during the search, some anti-cycling strategies are used to help the search process explore a broad portion of the solution space.

In this section, we design a TS algorithm for our problem. It is based on the general TS framework of (J. F. Cordeau, et al., 2001) with similar attribute set and augmented criterion function for constraint violations. The general framework of our TS is given in Algorithm 2. It starts with the best feasible solution of the GA initial population given in Section 4 and searches on the set of solutions in which each patient belongs to a route. The neighborhood is defined by relocation of a patient from one route to another route or location exchange of patients in different routes. Our tabu search also includes the re-optimization of all modified routes after each local move. The TS stops after L_1=5000 iterations or after L_2=2500 iterations without improvement of the best solution. Our TS restarts from the second best and empty tabu list if there is no improvement after L_3=1000 iterations. Restarts bring TS to new search regions (Dell'amico, et al., 1999).

In the following, we give detailed presentation of the augmented criterion function, the neighborhood structure, the route re-optimization, the attribute set, tabu duration and aspiration criterion.

Algorithm 2: Outline of the TS

```
1:  Determine an initial solution $Sol$;
2:  Determine the best neighbor solution $Sol'$ that is not tabu or satisfies an aspiration
    criterion;
```

```
3:  Re-optimize each modified route of Sol' and set Sol := Sol';
4:  Update tabu list and aspiration levels;
5:  Stop the TS if the stopping criterion is met;
6:  Restart the TS from the second best if a restart criterion is met;
7:  Go to 2.
```

5.1 Augmented criterion function

In our TS, both feasible and infeasible solutions are allowed. Each solution *Sol* partitions the set of patients into different routes and is completely represented by its set of routes *k* with each route represented by the sequence of nodes visited including the depot, the hospital and the lab.

For each route, the visiting times at different nodes can be easily determined by taking into account the visiting sequence and the earliest available time of each node. The vehicle load can be determined as follows. The vehicle starts at the depot with all demands of D_1 patients, drops all delivery demands d_{i1} and d_{i2} and loads all pickup demands p_{i1} and p_{i2} at each patient visit, loads all demands of D_2 patients of the route at the hospital, and drops all demands of P_2 patients of the route at the lab visit.

For each route *k* of a solution *Sol*, let $w_k(Sol)$ be the total traveling distance (original objective value), $d_k(Sol)$ be the total violation of vehicle capacity, $e_k(Sol)$ be the total violation of time window, $g_k(Sol)$ be the total violation of precedence constraints. The evaluation of $w_k(Sol)$ is straightforward and the evaluation of the others is as follows:

$$d_k(Sol) = \sum_{i\in R_k} \left(Load_{ik} - Q\right)^+ \tag{27}$$

$$e_k(Sol) = \sum_{i\in R_i} \left(m_{tk} - b_i\right)^+ \tag{28}$$

$$g_k(Sol) = N_{hk} + N_{lk} \tag{29}$$

where $(x)^+ = max(0, x)$, R_k is the set of nodes visited by vehicle *k*, $Load_{ik}$ is the vehicle load when leaving node *i*, m_{ik} is the visiting time of node *i* in route *k* and b_i its latest visiting time, N_{hk} (N_{lk}) is the number of D_2 (P_2) patients visited before a hospital visit (after a lab visit) in route *k*.

The following augmented criterion function *f* (*Sol*) is used in our TS:

$$f(Sol) = \sum_{k\in K} \left(w_k(Sol) + \alpha\cdot d_k(Sol) + \beta\cdot e_k(Sol) + \gamma\cdot g_k(Sol)\right) \tag{30}$$

where $\alpha, \beta,$ and γ are positive parameters to adjust the penalty of constraint violation. If *Sol* is a feasible solution, *f*(*Sol*) coincides with *w*(*Sol*). According to this definition, the TS search process will contain a mix of feasible and infeasible solutions, reducing the probability of becoming trapped in a local solution.

Our TS algorithm adjusts α, β, γ dynamically to facilitate the exploration of the search space. The TS algorithm starts or restarts from three initial parameters α_0, β_0 and γ_0, set at respectively 1, 1, 100. We set three intervals [$\alpha_{min}, \alpha_{max}$], [$\beta_{min}, \beta_{max}$] and [$\gamma_{min}, \gamma_{max}$] to limit these parameters during search process in the following respective ranges [0.01, 1000], [0.01, 1000] and [0.01, 5000]. At each TS iteration, penalty parameter for the vehicle capacity constraint α is modified as follows. If the solution generated after route re-optimization is feasible, α is divided by a factor 1+φ_1. If the solution is infeasible and the vehicle capacity is violated, α is multiplied by a factor 1+φ_1. If the solution is infeasible but the vehicle capacity is satisfied, parameter α is divided by a factor 1+φ_2 ($0 < \varphi_2 \le \varphi_1$). And, parameters β and γ are adjusted in the same rules. The following parameters φ_1 = 0.2 and φ_2 = 0.05 are used. Note these parameters, e.g., α, β, γ, φ_1 and φ_2, are tuned by the preliminary experiments, as well as some guidelines in previous studies on unified tabu search (J. Cordeau, et al., 1997). The sensitivity analyses on the parameters were performed sequentially, leaving the remaining parameters unchanged.

5.2 Neighborhood structure

Recall that each solution *Sol* in our TS partitions the set of patients into different routes and is completely represented by its set of routes with each route represented by the sequence of nodes visited including the depot, the hospital and the lab.

The neighborhood of a solution *Sol* is defined by two local moves. The first local move for a solution *Sol* removes a patient from one route *k* and inserts it into another nonempty route *k′*. The second local move consists in exchanging the route assignment of two patients *i* and *j* in different routes *k* and *k′*. The relocation move can be denoted as (*i*, *k*') and route exchange move denoted as (*i*, *j*, *k*, *k*')

More specifically, relocation move (*i*, *k*') of a patient on route *k* consists in (i) removing patient *i* from route *k*; (ii) inserting it in route *k′* at a position that minimize the cost of route *k′*, i.e.

$$w_{k'}(Sol) + \alpha \cdot d_{k'}(Sol) + \beta \cdot e_{k'}(Sol) + \gamma \cdot g_{k'}(Sol),$$

and (iii) removing the hospital (lab) in route *k* if there is no more D_2 (P_2) patient in route *k* after the local move. Route exchange move (*i*, *j*, *k*, *k*') of two patients on different routes *k* and *k*' consists in (i) removing patients *i* and *j* from their routes, (ii) inserting patient *i* (*j*) in route *k*' (*k*) at a position that minimize the cost of route *k*' (*k*'), and (iii) removing any unnecessary hospital and lab visit in routes *k* and *k*'. The cost of the neighbor solution is the augmented cost function of the solution obtained after the local move.

Note that when inserting a patient in route *k*, we do not relocate or insert the hospital visit and the lab visit in the route even if it is necessary to satisfy the new patient. This is due to the relative long computation time needed to find the best position for inserting hospital or lab. Nevertheless, the relocation or insertion of hospital and lab visits are considered in the re-optimization of modified route once the next local move has been selected.

Other local moves such as *intra-route crossover* method and 2-*opt** (Alfredo Tang Montané and Galvão, 2006) were also used in the literature for solving the VRPs and VRPSDP. We tested such local moves in our TS and did not observe significant improvement. Since most of our TS running time is spent by the neighborhood search, implement more local moves has a clear disadvantage of increasing very substantially the size of the neighborhood and the computation time. For this reason, we limit our TS to two local moves.

5.3 Re-optimization of a modified route

Each local move in our TS can modify two routes. A modified route might miss necessary hospital or lab visit, violate vehicle capacity, time window and precedence constraints. This subsection presents a method to rebuild and improve each modified route. The remaining part of this subsection concerns the re-optimization of a given modified route.

The re-optimization starts with a constructive method and then improves the route by local search. More specifically, it starts with a simple and effective constructive heuristic, the *nearest insertion* (*NI*) (Bentley, 1992). Our *NI* heuristic is a three-step method.

Step1: Delete all the existing trips in this route. Then create a partial route beginning and ending at the depot and visiting the farthest patient.

Step2: Insert the patient whose insertion generates the smallest increment on the cost $w(Sol)+\alpha \cdot d(Sol)+\beta \cdot e(Sol)$, i.e., travel cost plus penalty of the vehicle capacity and time window violation. This step is repeated until all patients are inserted.

Step3: Insert the lab and hospital into the route at feasible positions with the smallest increment on the cost $w(Sol)+\alpha \cdot d(Sol)+\beta \cdot e(Sol)$. Precedence constraints are satisfied and $g(Sol)$=0 at this step, i.e. the hospital is visited before all D_2 patients and the lab is visited after all P_2 patients.

The *NI* route is not necessarily good enough or even feasible for our problem. It is further improved by local search combing the first-accept strategy and three moves: 1-1exchange, 1-0 relocation, 2-Opt exchange. Before the local search, we check the feasibility of the *NI* route. If the *NI* route is feasible, infeasible solutions are not allowed in local search. If the *NI* route is not feasible, infeasible neighbor solutions are allowed in local search and the augmented criterion function *f(Sol)* is used.

5.4 Attribute set, Tabu list, Tabu duration and Aspiration criterion

In our TS, we associate with each solution *Sol* an attribute set $At(Sol)=\{(i, k)|i\in N, k\in K\}$ indicating for each patient *i* the vehicle *k* serving it. Of course, the attribution set is only a partial characterization of the solution used to define the tabu list. The relocation move (i, k') of a patient *i* in route *k* to a different route *k'* is equivalent to replace an attribute (i, k) from $At(Sol)$ by a new attribute (i, k') with $k \neq k'$. The route exchange of two patients in different routes can be seen as replacing two attributes in $At(Sol)$.

TS utilizes adaptive memory, called tabu list and tabu duration, to implement a diversification strategy. In our TS, when a patient *i* is removed from a route *k*, we assign a tabu status to the attribute (*i*, *k*), and set a tabu duration θ to this attribute. That is to say, in the next θ iterations, inserting patient *i* back into route *k* is forbidden when we perform the neighborhood search. The size of the tabu list θ takes its values in $[\theta_{\min}, \theta_{\max}]$ and starts from θ_0. Tabu list θ is also a self-adjusting parameter, and dynamically modifies during the TS search. After each improvement of the current best solution S_{best}, we set parameter θ equal to θ_{min}. After θ_ϕ consecutive times unimproved iteration, parameter θ is updated to be $min(\theta+1, \theta_{max})$. In the preliminary experiments, we find θ_0 = 7, θ_ϕ = 30 suit most test instances. It is not necessary to increase the value of θ_0 with the instance size (customer number). And, we set $[\theta_{\min}, \theta_{\max}]$ = [5, 15] in our TS implementation.

The tabu status of a local move can be overridden by an aspiration criterion in our TS. We define an aspiration value for each attribute, which is equal to the cost of the best feasible solution found with that attribute. For each feasible solution found after the re-optimization of modified routes, we update the aspiration value of each attribute of this feasible solution. A relocation move (*i*, *k*') is considered at an iteration if (*i*, *k*') is not in the tabu list, or the neighbor solution is feasible and its cost is smaller than the aspiration value of (*i*, *k*'). A route exchange move (*i*, *j*, *k*, *k*') is considered if (*i*, *k*') and (*j*, *k*) are not in the tabu list, or the neighbor solution is feasible and its cost is smaller than the smallest aspiration value of (*i*, *k*') and (*j*, *k*).

6 Computational experiments

This section reports the results of a series of computational experiments for comparison of the genetic algorithm, the tabu search, and application of the commercial solver Cplex 12.3 for the two mathematical formulations MIP1 and MIP2 of Section 3, and other methods of the literature for some special cases of our problem.

To the best of our knowledge, this paper is the first study of this special vehicle scheduling problem in the home health care industry. There are no benchmark instances to evaluate the performances of our heuristic approaches. Therefore, we construct some test instances based on existing VRPTW benchmarks. Further, as our problem with only P_1 or D_1 patients reduces to the classic VRPMBTW, our approaches are also compared with existing VRPMBTW approaches on existing benchmark instances for the VRPMBTW.

As our problem is highly combinatorial, the performance of Cplex solver strongly depends on its parameterization. We tried different Cplex parameters including default settings, strong branching, depth-first search, MIP emphasis feasibility or optimality. No parameter setting led to satisfactory performance. For most of the small-size test instances of this Section, Cplex cannot find a feasible solution after 48 hours. It does not make sense to compare our heuristic solutions with direct implementation of MIP models in Cplex. Instead, we use in this section *MIPstart* strategy by letting Cplex to start with the initial solution of our TS, i.e. the best initial solution of our GA. Cplex installs it as the incumbent and initial solution of its branch and cut procedure, which allows Cplex to eliminate portions of the search space and results in smaller branch and cut trees. Cplex starts from this initial solution and goes on solving the problem based on this solution, until exhausting the memory or predetermined maximum computation time. In our preliminary experiments, we also find that Cplex with MIP2 formulation always outperforms Cplex with MIP1. For this reason, this section limits to Cplex with MIP2 formulation. Further, the Cplex lower bounds are very poor and are not given in this paper.

All the algorithms of this paper are implemented in C. All heuristic algorithms (GA, TS,…) are carried out on a 3.2 GHz Dual Core computer with a 2 GB memory under Linux. We set a time limit of 72 hours and a memory limit of 20 GB for Cplex for each instance. After some preliminary experiments, the parameters of the proposed GA and TS have been set to the values reported earlier in this paper. For fair comparison with GA, our tabu search algorithm is implemented with two stopping criteria: (i) maximum number of iterations and maximum number of iterations without improvement, and (ii) same computation time as GA. The first tabu search is denoted TS1 and the second one TS2. GA, TS1 and TS2 run 10 times for each instance. The best results, the average results and average running time are used to assess the efficiency of these algorithms.

6.1 Test Instances from VRPTW benchmarks

We first derive test instances from existing VRPTW benchmarks of (Solomon, 1987) and (Gehring and Homberger, 1999). Eighteen Solomon VRPTW instances are selected to generate our test instances. Each Solomon instance contains 100 customers over a service region defined on a 100×100 grid. These VRPTW instances are divided into three classes that differ by the geographical distribution of the customers: they are clustered in the *C* type instances, randomly located in the *R* type instances, and partly clustered, partly randomly located in the *RC* type instances. Meanwhile, each class is divided into two series: in the 100-series instances time windows are tighter, and in the 200-series instances time windows are wider. To test different characteristics of instances, 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.

For each Solomon instance, we derive 6 new instances for our problem with 4Z demands as follows. First, we randomly choose Z customers from the Solomon instance as the P_1 patients in our new instance, each of which has a demand equal to 50% of the customer's demand given in the Solomon instance. Then, Z P_2, Z D_1 and Z D_2 patients are randomly selected from the Solomon instance. Clearly, one patient may be selected more than once and the number of patients is less than then number of requirements (4Z). If the basic Solomon instance is C101 and 37 patients and 40 demands exist in our new instance, it is denoted as C101-37-40. The coordinates of the depot is inherited in our instances, and the locations of lab and hospital are (10, 15) and (40, 50). For each patient, the time window in the Solomon instance is used directly. Time windows for the depot, lab and hospital are selected as follows to avoidance infeasible solutions. The depot’s time window of the Solomon instance is multiplied by 1.2 and assigned to the depot, lab and hospital in our instance. If ‘*violative’* patients still exist in our instance, the new time window is repeatedly multiplied again by 1.2 until all ‘*violative’* patients are eliminated. In the preliminary experiment, we find that all ‘*violative’* patients disappear after two tries.

For each Solomon instance, this constructing procedure are repeated 6 times, generating 2 small (40 demands), 2 moderate (80 demands), and 2 large (120 demands) instances. Concerning the vehicle capacity and vehicle number, their values have been reduced compared to the ones considered for the VRPTW, because they are loose for our problem. The detailed information about these two characteristics is illustrated in Tables 4-7.

Besides the Solomon instances, we also create the largest instances from VRPTW instances of (Gehring and Homberger, 1999). These instances are similar but larger than Solomon instances and have hundreds of customers. We choose 12 instances from this benchmark, each of which has 400 customers, and undergoes the procedure described above to generate 24 new instances for our study. Each of these instances contains 200 demands, and also named as the VRPTW instance’s label with patients number, and demands number.

6.2 Computational Results on VRPTW-based instances

In this subsection, Tables 1 and 2 summarize the results obtained from the GA and TS on all VRPTW-based instances. The detailed computational results obtained on the small instances (containing 40 demands), moderate instances (80 demands), large instances (120 demands), and the largest instances (200 demands) are presented in four Tables 4-7 in the Appendix, respectively.

Table 1. Average routing costs of 10 independent GA and TS runs on VRPTW-based instances

Demand #	*Type*	*Cplex*	*GA*					*TS1*					*TS2*			
			Best	*AVG*	*Worst*	*STD*	*CPU*	*Best*	*AVG*	*Worst*	*STD*	*CPU*	*Best*	*AVG*	*Worst*	*STD*
	C	1065.7	881.9	890.5	895.6	5.6	88.6	882.0	893.3	900.7	7.0	63.6	881.6	890.1	899.1	6.1
	R	1051.0	875.3	889.0	901.2	10.1	74.2	873.5	884.2	894.1	6.8	52.6	874.1	882.5	889.5	6.6
50	RC	1158.3	954.4	966.4	977.2	7.1	78.0	955.3	970.6	983.3	8.5	67.3	955.4	965.3	972.8	6.0
	C	1776.8	1424.3	1457.4	1480.7	18.4	356.0	1423.0	1463.2	1481.1	18.9	335.6	1419.7	1466.2	1491.5	20.3
	R	1680.7	1409.1	1441.1	1463.9	24.4	343.4	1410.2	1446.6	1467.4	22.0	243.3	1402.1	1436.4	1455.5	17.5
80	RC	1867.8	1577.6	1617.5	1648.9	20.6	298.6	1572.1	1610.7	1634.2	19.3	218.6	1570.9	1612.8	1641.2	20.4
	C	1965.6	1503.9	1542.1	1565.1	27.7	615.3	1514.9	1565.0	1599.1	26.9	597.2	1502.0	1550.4	1584.0	26.3
	R	2012.8	1505.4	1553.5	1575.9	27.1	617.3	1514.4	1566.8	1601.1	28.1	558.2	1513.9	1558.4	1584.8	25.2
120	RC	2181.7	1691.3	1736.4	1764.8	25.7	541.9	1692.6	1728.2	1764.5	25.6	471.1	1682.3	1726.8	1753.8	24.3
	C	9744.1	8267.5	8387.2	8485.5	54.7	3274.4	8166.7	8283.9	8349.6	53.6	4301.6	8198.7	8295.2	8369.3	53.8
	R	10759.9	9219.3	9380.8	9499.1	54.0	3111.0	9107.4	9239.7	9305.1	56.9	3707.8	9091.4	9164.7	9234.7	58.7
200	RC	9773.1	8580.1	8706.5	8808.7	62.7	3342.7	8456.1	8548.9	8622.3	54.0	3784.9	8473.1	8555.2	8624.7	55.6
AVG		3176.8	2654.6	2704.0	2739.6	25.5	863.5	2635.6	2682.7	2713.0	24.8	951.8	2634.3	2675.7	2705.3	24.1

Table 2. Percentage of best solutions found of GA and TS on VRPTW-based instances

Demand#	*Type*	*Cplex*	*GA*	*TS1*	*TS2*
40	C	0%	83%	92%	92%
	R	0%	58%	67%	58%
	RC	0%	75%	75%	67%
80	C	0%	42%	42%	58%
	R	0%	42%	17%	58%
	RC	0%	50%	25%	42%
120	C	0%	50%	42%	67%
	R	0%	58%	17%	25%
	RC	0%	42%	25%	42%
200	C	0%	13%	63%	38%
	R	0%	25%	38%	50%
	RC	0%	13%	63%	25%
AVG		0%	48%	46%	53%

Table 1 shows the average routing costs by grouping problem instances according to the number of demands and the type of the instance. The results are obtained from 10 independent runs for each problem instance of the three approaches (GA, TS1, and TS2) plus the one obtained with CPLEX. Column '*Best*' is the average over all relevant problem instances of the best solutions among 10 independent runs of an approach for each instance. Column '*AVG*' is the average travel cost over all problem instances and over all runs of the approach. Similarly, column '*Worst*' represents the average value of the worst solution costs among relevant test instances. We calculate the standard deviation of 10 runs of each test instance, and Column '*STD*' gives the average standard deviation among all relevant test instances. Column '*CPU*' is the average CPU time in seconds of one run among relevant test instances. The last line in Table 1 provides the average values for all the test instances. Table 2 shows the percentage of the best solutions obtained by GA, TS1 and TS2 on each type of test instances.

Several conclusions can be drawn from these experimental results. First, the proposed GA and TS (both TS1 and TS2) perform well for test instances of different types and different sizes. For each combination of types and sizes, our heuristic algorithms significantly dominate the Cplex solver. For all 132 test instances, the Cplex solution costs and best GA solution costs deviate on average by 16.4%; the Cplex solution costs and TS1, TS2 best solution costs deviate on average up to 17.0% and 17.1%, respectively. Recall that both Cplex and TS start from the best solution of the initial GA population built by simple constructive heuristics, random generated solutions and the optimal split procedure. This implies that GA and TS can significantly improve the solutions of these heuristic solutions and our algorithmic approaches are highly competitive.

Concerning the solution quality of the GA and TS approaches, the performances of all three heuristics are satisfactory. For example, as shown in Table 2, out of 36 small size test instances, GA, TS1 and TS2 can get 72%, 78%, 72% best solutions, respectively. For 17 instances out of all 36 small size instances, GA, TS1 and TS2 all find the same solution (referred to Tables in Appendix). Among the total 132 test instances, GA, TS1 and TS2 are able to find 48%, 46%, 53% best solutions, respectively. Note that Cplex does not get any best solution and the Cplex solution is always far from the best solutions found by other approaches.

Meanwhile, we find that our approaches GA, TS1 and TS2 are robust and can find good solutions for different test instances in different runs. As shown in Table 1, for all the test instances, the average standard deviation of GA, TS1 and TS2 are 25.5, 24.8 and 24.1. Comparing GA and TS, we find that TS1 is slightly better the GA. For the total 132 test instances, the best solution costs of GA and TS1 deviate on average by 0.71%; their average and worst solution costs deviate by 0.79% and 0.97%, respectively. The superiority of TS1 is especially true for problem instances of the largest size. For example, as shown in Table 1, for the largest 24 instances, the best solution costs of GA and TS1 deviate on average by 1.29%; the deviation between their average and worst solution costs is 1.52% and 1.93%. Meanwhile, Table 2 shows that TS1 succeeds in finding 55% best solutions of the largest instances, while GA is able to find 17% best solutions. Nevertheless, for these instances, TS1's improved solution quality is obtained at the cost of longer computation time. Over the 132 test instances, the CPU time of TS1 is on average 9.28% longer than that of GA.

With the same computation time between GA and TS2, the performance of TS2 is slightly better than that of the GA. Over 132 test instances, concerning their best solution, TS2 is slightly better than GA on the average best solution costs with a deviation of 0.76%. Meanwhile, TS2 can find more best solutions than GA, with 48% best solutions found by GA and 53% by TS2.

6.3 Test on VRPMBTW benchmarks

This subsection restrict our problem to only P_1 and D_1 patients and our problem reduces to the classic *VRP with mixed backhauls and time windows* (*VRPMBTW*). Our GA and TS are tested on the benchmarks for the VRPMBTW against existing best solutions. (Gélinas, et al., 1995) construct test data for the basic VRPBTW (all backhaul customers are serviced after all the linehaul customers) from Solomon VRPTW problems. They constructed VRPBTW instances by randomly choosing 10%, 30% and 50% of the 100 total customers in Solomon VRPTW benchmark to be backhaul customers instead. (Hasama, et al., 1998) designed 15 VRPMBTW test problems from the data of (Gélinas, et al., 1995) by relaxing the linehaul-before-backhaul constraint, and proposed a simulated annealing (SA) heuristic method for the VRPMBTW. We test our two heuristics, GA and TS2, on Hasama's problem instances.

The results obtained by our algorithms and those provided the SA algorithm of (Hasama, et al., 1998) are detailed in Table 3. Column '*Name*' gives the label of the instance. Column '*BH%*' presents the percentage of the backhaul among all customers. Columns 3 and 4 contain the best solutions found by Hasama's SA algorithm, where '*Cost*' shows the travel distance and '*V*' shows the number of the vehicles used. The next six columns contain the best, the average and the worst solution cost, the standard deviation, the average CPU time (for one run) and minimum vehicle number used in 10 GA runs for each instance. The number of vehicles used in the minimum cost solution is also given in parenthesis if it is larger than *V*. Similar results are given for TS2.

Table 3. Results on VRPMBTW test instances

Instance		*SA*		*GA*						*TS2*				
Name	*BH%*	*Cost*	*V*	*Best*	*AVG*	*Worst*	*STD*	*CPU*	*V*	*Best*	*AVG*	*Worst*	*STD*	*V*
R101	10	1686.5	**21**	1685.0	1694.6	1705.5	7.4	704.1	**21**	**1680.1**	1683.1	1685.4	3.6	**21**
R101	30	1693.4	**20**	1667.4 (22)	1686.0	1709.9	10.7	578.2	21	**1663.5**	1677.1	1668.9	5.0	21
R101	50	1696.4	**21**	**1664.0**	1691.2	1711.1	14.7	717.0	**21**	1665.8	1687.0	1699.3	10.8	**21**
R102	10	1753.4	20	**1490.3**	1499.7	1521.9	13.2	702.9	19	1494.1(19)	1501.2	1508.6	6.2	**18**
R102	30	1583.0	19	1492.6	1510.4	1523.5	14.4	730.2	**18**	**1489.8**	1518.4	1530.7	13.9	**18**
R102	50	1602.7	**19**	**1486.8**	1499.6	1509.0	12.9	581.3	**19**	**1486.8**	1507.1	1519.0	7.4	**19**
R103	10	1250.6	**15**	1228.6	1242.0	1251.4	10.7	720.5	**15**	**1228.4**	1247.2	1259.8	12.4	**15**
R103	30	1259.1	16	1224.7	1240.6	1260.5	12.1	596.4	**15**	**1223.8**	1229.9	1231.1	2.6	**15**
R103	50	1495.5	17	**1227.9**	1240.6	1253.9	7.7	693.5	**15**	1231.0	1237.0	1241.4	2.9	**15**
R104	10	1078.2	**11**	**1002.5**	1012.0	1022.1	8.8	623.9	12	1006.6(12)	1015.2	1022.9	6.1	**11**
R104	30	1125.3	12	1000.6 (12)	1020.8	1041.9	13.9	759.3	**11**	**1000.2**(12)	1012.6	1020.0	4.2	**11**
R104	50	1177.4	12	**1002.5**	1024.0	1038.5	12.7	729.8	12	1006.8	1017.7	1025.2	6.9	**11**
R105	10	1479.4	**16**	**1394.6** (17)	1413.3	1435.2	12.6	586.5	**16**	1411.2	1419.3	1426.1	3.0	**16**
R105	30	1417.4	15	**1415.8**	1432.1	1450.5	11.4	809.7	17	1416.0	1432.9	1442.5	8.9	17
R105	50	1464.5	15	1396.5	1427.2	1442.9	13.0	703.3	16	**1396.2**	1416.5	1428.4	7.0	16
Average		1450.9		1358.7	1375.6	1391.9	11.7	682.4		1360.0	1373.5	1380.6	6.7	16.3

From Table 3, our GA and TS2 algorithms dominate the SA algorithm of (Hasama, et al., 1998) in terms of solution quality. The primary objective of (Hasama, et al., 1998) is to minimize the number of vehicles used to serve the customers, and the second objective is to minimize the total travel distance. Therefore, we first compare the number of vehicles. Although our heuristics **do not** use this as their objective, the proposed GA still can reduces the number of vehicles needed for 4 out of the 15 instances, and TS2 reduces the number of vehicles for 6 instances. Concerning the travel distance, the solutions of our GA and TS2 are much better than the solution produced by (Hasama, et al., 1998). For each instance, GA and TS2 can get better solution. The average percentage deviations between the best solution costs of our GA and TS2 and Hasama's SA are 6.35% and 6.27% respectively.

7 Conclusions and future research

This paper investigates a special simultaneous pickup and delivery problem with time windows in HHC, an extension of the classical VRPSDPTW. The problem is of interest because of its theoretical complexity and of the important applications in the home health care industry. We formulate the problem as two integer programming models to minimize the total vehicle cost for serving all patients demands. We also propose two meta-heuristics, Tabu search and genetic algorithm, to solve this problem. As this problem is new and no benchmark exists, experiments are conducted by using a range of test instances, which are designed based on existing VRPTW benchmarks to reflect different realistic scenarios. In general, both TS and GA can provide good solutions in a reasonable time span, and TS requires relatively more computational time. Our proposed heuristic approaches are also tested on a set of VRPMBTW benchmarks against best known results. The results of our metaheuristics are clearly better than the best-known solutions in the existing literature.

This research can be extended in different directions. Firstly, the problem can be extended to a planning horizon of several days, e.g., a week, to combine planning of delivery/pickup and vehicle routing. In real-life applications, usually each patient requires a certain number of visits and services within this time horizon. The HHC has to choose the visiting days for each patient and to solve a vehicle scheduling problem for each day. In the vehicle scheduling problem of each day, some special constraints and conditions encountered in the HHC must be considered. This problem is a special periodic vehicle scheduling problem and even more complex than the problem studied in this paper. Meanwhile, in many real-world home health care logistic applications, some elements in the problem may not be known in advance, e.g., patients' delivery/pickup demands, travel and service times of the vehicle. Such uncertain elements significantly affect the system performance. The vehicle scheduling problems in HHC under such uncertain conditions are also very important and interesting.

Acknowledgement

This research is partially funded by Saint Etienne Metropôle for providing a Post-Doc fellowship for the first author. The authors are also grateful to OIKIA home health care company for sharing information for home health care logistics.

References

Ai, T. J., Kachitvichyanukul, V. (2009). A particle swarm optimization for the vehicle routing problem with simultaneous pickup and delivery. *Computers & Operations Research, 36*, 1693-1702.

Alfredo Tang Montané, F., Galvão, R. D. (2006). A tabu search algorithm for the vehicle routing problem with simultaneous pick-up and delivery service. *Computers & Operations Research, 33*, 595-619.

Angelelli, E., Mansini, R. (2002). The vehicle routing problem with time windows and simultaneous pick-up and delivery. *Quantitative approaches to distribution logistics and supply chain management*, 249–267.

Baldacci, R., Hadjiconstantinou, E., Mingozzi, A. (2003). An exact algorithm for the traveling salesman problem with deliveries and collections. *Networks, 42*, 26-41.

Bentley, J. J. (1992). Fast algorithms for geometric traveling salesman problems. *Informs Journal on Computing, 4*, 387-411.

Berbeglia, G., Cordeau, J. F., Gribkovskaia, I., Laporte, G. (2007). Static pickup and delivery problems: a classification scheme and survey. *Top, 15*, 1-31.

Bianchessi, N., Righini, G. (2007). Heuristic algorithms for the vehicle routing problem with simultaneous pick-up and delivery. *Computers & Operations Research, 34*, 578-594.

Côté, J.-F., Potvin, J.-Y. (2009). A tabu search heuristic for the vehicle routing problem with private fleet and common carrier. *European Journal of Operational Research, 198*, 464-469.

Chen, J. F., Wu, T. H. (2005). Vehicle routing problem with simultaneous deliveries and pickups. *Journal of the Operational Research Society, 57*, 579-587.

Cordeau, J.-F. (2006). A Branch-and-Cut Algorithm for the Dial-a-Ride Problem. *Operations Research, 54*, 573-586.

Cordeau, J.-F., Laporte, G. (2003). A tabu search heuristic for the static multi-vehicle dial-a-ride problem. *Transportation Research Part B: Methodological, 37*, 579-594.

Cordeau, J., Gendreau, M., Laporte, G. (1997). A tabu search heuristic for periodic and multi-depot vehicle routing problems. *Networks, 30*, 105-119.

Cordeau, J. F., Laporte, G., Mercier, A. (2001). A unified tabu search heuristic for vehicle routing problems with time windows. *Journal of the Operational Research Society, 52*, 928-936.

Crispim, J., Brandão, J. (2005). Metaheuristics applied to mixed and simultaneous extensions of vehicle routing problems with backhauls. *Journal of the Operational Research Society, 56*, 1296-1302.

Dell'amico, M., Lodi, A., Maffioli, F. (1999). Solution of the Cumulative Assignment Problem With a Well-Structured Tabu Search Method. *Journal of Heuristics, 5*, 123-143.

Dell’Amico, M., Righini, G., Salani, M. (2006). A Branch-and-Price Approach to the Vehicle Routing Problem with Simultaneous Distribution and Collection. *Transportation Science, 40*, 235-247.

Dethloff, J. (2002). Relation between vehicle routing problems: an insertion heuristic for the vehicle routing problem with simultaneous delivery and pick-up applied to the vehicle routing problem with backhauls. *Journal of the Operational Research Society, 53*, 115-118.

Diana, M., Dessouky, M. M. (2004). A new regret insertion heuristic for solving large-scale dial-a-ride problems with time windows. *Transportation Research Part B: Methodological, 38*, 539-557.

Duhamel, C., Potvin, J. Y., Rousseau, J. M. (1997). A tabu search heuristic for the vehicle routing problem with backhauls and time windows. *Transportation Science, 31*, 49-59.

Dumas, Y., Desrosiers, J., Soumis, F. (1991). The pickup and delivery problem with time windows. *European Journal of Operational Research, 54*, 7-22.

Eilam Tzoreff, T., Granot, D., Granot, F., Sošić, G. (2002). The vehicle routing problem with pickups and deliveries on some special graphs. *Discrete Applied Mathematics, 116*, 193-229.

Gélinas, S., Desrochers, M., Desrosiers, J., Solomon, M. M. (1995). A new branching strategy for time constrained routing problems with application to backhauling. *Annals of Operations Research, 61*, 91-109.

Gajpal, Y., Abad, P. (2009). An ant colony system (ACS) for vehicle routing problem with simultaneous delivery and pickup. *Computers & Operations Research, 36*, 3215-3223.

Gajpal, Y., Abad, P. (2009). Multi-ant colony system (MACS) for a vehicle routing problem with backhauls. *European Journal of Operational Research, 196*, 102-117.

Gehring, H., Homberger, J. (1999). A parallel hybrid evolutionary metaheuristic for the vehicle routing problem with time windows. In.

Gendreau, M., Hertz, A., Laporte, G. (1994). A Tabu Search Heuristic for the Vehicle Routing Problem. *Management Science, 40*, 1276-1290.

Goetschalckx, M., Jacobs-Blecha, C. (1989). The vehicle routing problem with backhauls. *European Journal of Operational Research, 42*, 39-51.

Hasama, T., Kokubugata, H., Kawashima, H. (1998). A heuristic approach based on the string model to solve vehicle routing problem with backhauls. In.

Hertz, A., Laporte, G., Mittaz, M. (2000). A tabu search heuristic for the capacitated arc routing problem. *Operations Research, 48*, 129-135.

Hokey, M. (1989). The multiple vehicle routing problem with simultaneous delivery and pick-up points. *Transportation Research Part A: General, 23*, 377-386.

Jaw, J. J., Odoni, A. R., Psaraftis, H. N., Wilson, N. H. M. (1986). A heuristic algorithm for the multi-vehicle advance request dial-a-ride problem with time windows. *Transportation Research Part B: Methodological, 20*, 243-257.

Kontoravdis, G., Bard, J. F. (1995). A GRASP for the vehicle routing problem with time windows. *ORSA journal on Computing, 7*, 10-10.

Li, H., Lim, A. (2001). A Metaheuristic for the Pickup and Delivery Problem with Time Windows. In *Proceedings of the 13th IEEE International Conference on Tools with Artificial Intelligence* (pp. 160): IEEE Computer Society.

Lu, Q., Dessouky, M. (2006). A new insertion-based construction heuristic for solving the pickup and delivery problem with time windows. *European Journal of Operational Research, 175*, 672-687.

Madsen, O. B. G., Ravn, H. F., Rygaard, J. M. (1995). A heuristic algorithm for a dial-a-ride problem with time windows, multiple capacities, and multiple objectives. *Annals of Operations Research, 60,* 193-208.

Mingozzi, A., Giorgi, S., Baldacci, R. (1999). An exact method for the vehicle routing problem with backhauls. *Transportation Science, 33*, 315-329.

Mingyong, L., Erbao, C. (2010). An improved differential evolution algorithm for vehicle routing problem with simultaneous pickups and deliveries and time windows. *Engineering Applications of Artificial Intelligence, 23*, 188-195.

Nagy, G., Salhi, S. (2005). Heuristic algorithms for single and multiple depot vehicle routing problems with pickups and deliveries. *European Journal of Operational Research, 162*, 126-141.

Nanry, W. P., Wesley Barnes, J. (2000). Solving the pickup and delivery problem with time windows using reactive tabu search. *Transportation Research Part B: Methodological, 34*, 107-121.

Osman, I. H., Wassan, N. A. (2002). A reactive tabu search meta‐heuristic for the vehicle routing problem with back‐hauls. *Journal of Scheduling, 5*, 263-285.

Pankratz, G. (2005). A grouping genetic algorithm for the pickup and delivery problem with time windows. *Or Spectrum, 27*, 21-41.

Parragh, S., Doerner, K., Hartl, R. (2008). A survey on pickup and delivery problems Part II: Transportation between pickup and delivery locations. *Journal für Betriebswirtschaft, 58*, 81-117.

Parragh, S. N., Doerner, K. F., Hartl, R. F. (2010). Variable neighborhood search for the dial-a-ride problem. *Computers & Operations Research, 37*, 1129-1138.

Prins, C. (2004). A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers & Operations Research, 31*, 1985-2002.

REIMANN, M., DOERNER, K., HARTL, R. F. (2002). Insertion based ants for Vehicle Routing problems with Backhauls and time windows. *Lecture Notes in Computer Science*, 135-148.

Reimann, M., Ulrich, H. (2006). Comparing backhauling strategies in vehicle routing using ant colony optimization. *Central European Journal of Operations Research, 14*, 105-123.

Ropke, S., Cordeau, J. F. (2009). Branch and cut and price for the pickup and delivery problem with time windows. *Transportation Science, 43*, 267-286.

Ropke, S., Cordeau, J. F., Laporte, G. (2007). Models and branch‐and‐cut algorithms for pickup and delivery problems with time windows. *Networks, 49*, 258-272.

Ropke, S., Pisinger, D. (2006). An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows. *Transportation Science, 40*, 455-472.

Süral, H., Bookbinder, J. H. (2003). The single‐vehicle routing problem with unrestricted backhauls. *Networks, 41*, 127-136.

Sörensen, K., Sevaux, M. (2006). MA|PM: Memetic algorithms with population management. *Computers & Operations Research, 33,* 1214-1225.

Salhi, S., Nagy, G. (1999). A cluster insertion heuristic for single and multiple depot vehicle routing problems with backhauling. *The Journal of the Operational Research Society, 50,* 1034-1042.

Savelsbergh, M., Sol, M. (1998). DRIVE: Dynamic routing of independent vehicles. *Operations Research, 46*, 474-490.

Savelsbergh, M. W. P., Sol, M. (1995). The general pickup and delivery problem. *Transportation Science, 29*, 17-29.

Sigurd, M., Pisinger, D. (2004). Scheduling transportation of live animals to avoid the spread of diseases. *Transportation Science, 38*, 197-209.

Solomon, M. M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research, 35*, 254-265.

Subramanian, A., Drummond, L., Bentes, C., Ochi, L., Farias, R. (2010). A parallel heuristic for the vehicle routing problem with simultaneous pickup and delivery. *Computers & Operations Research, 37*, 1899-1911.

Tavakkoli-Moghaddam, R., Saremi, A., Ziaee, M. (2006). A memetic algorithm for a vehicle routing problem with backhauls. *Applied Mathematics and Computation, 181*, 1049-1060.

Thangiah, S. R., Potvin, J. Y., Sun, T. (1996). Heuristic approaches to vehicle routing with backhauls and time windows. *Computers & Operations Research, 23*, 1043-1057.

Toth, P., Vigo, D. (1997a). An exact algorithm for the vehicle routing problem with backhauls. *Transportation Science, 31*, 372-385.

Toth, P., Vigo, D. (1997b). Heuristic algorithms for the handicapped persons transportation problem. *Transportation Science, 31*, 60-71.

Toth, P., Vigo, D. (1999). A heuristic algorithm for the symmetric and asymmetric vehicle routing problems with backhauls. *European Journal of Operational Research, 113*, 528-543.

Wade, A., Salhi, S. (2004). An ant system algorithm for the mixed vehicle routing problem with backhauls. In (pp. 699-719): Kluwer Academic Publishers.

Wang, H.-F., Chen, Y.-Y. (2012). A genetic algorithm for the simultaneous delivery and pickup problems with time window. *Computers & Industrial Engineering, 62*, 84-95.

Woodward, C. A., Abelson, J., Tedford, S., Hutchison, B. (2004). What is important to continuity in home care?: Perspectives of key stakeholders. *Social Science &amp; Medicine, 58*, 177-192.

Xu, H., Chen, Z. L., Rajagopal, S., Arunapuram, S. (2003). Solving a practical pickup and delivery problem. *Transportation Science, 37*, 347-364.

Yano, C. A., Chan, T. J., Richter, L. K., Cutler, T., Murty, K. G., McGettigan, D. (1987). Vehicle routing at quality stores. *Interfaces, 17*, 52-63.

Zachariadis, E. E., Tarantilis, C. D., Kiranoudis, C. T. (2009). A Guided Tabu Search for the Vehicle Routing Problem with two-dimensional loading constraints. *European Journal of Operational Research, 195*, 729-743.

Zhong, Y., Cole, M. H. (2005). A vehicle routing problem with backhauls and time windows: a guided local search solution. *Transportation Research Part E: Logistics and Transportation Review, 41*, 131-144.