Vehicle routing problems with alternative paths: an application to on-demand transportation

Thierry Garaix, Christian Artigues, Dominique Feillet, Didier Josselin

Vehicle routing problems with alternative paths: an application to on-demand transportation

Thierry GARAIX[♠♣], Christian ARTIGUES[♢],
Dominique FEILLET[♣], Didier JOSSELIN[♠]
[♣]Laboratoire d'Informatique d'Avignon,
339 Chemin des Meinajariés, BP 1228, 84911 Avignon, France
[♢]LAAS, CNRS 7 Avenue du Colonel Roche 31077 Toulouse, France
[♠]UMR ESPACE 6012 CNRS, Université d'Avignon,
74 rue Louis Pasteur, 84029 Avignon, France

April 22, 2007

Abstract

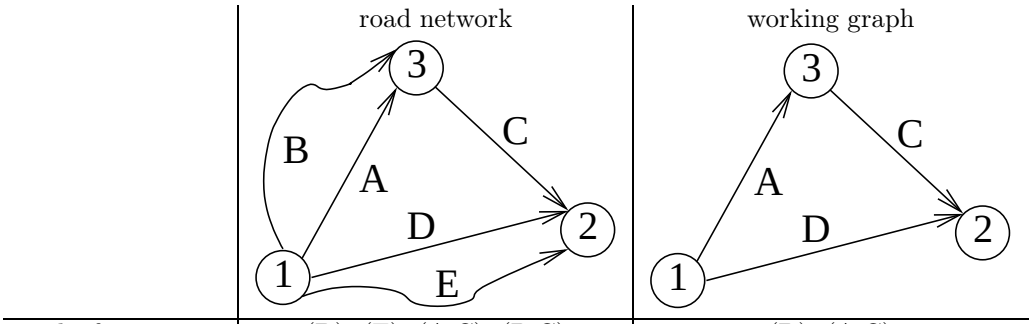
The class of vehicle routing problems involves the optimization of freight or passenger transportation activities. These problems are generally treated via the representation of the road network as a weighted complete graph. Each arc of the graph represents the shortest route for a possible origin-destination connection. Several attributes can be defined for one arc (travel time, travel cost ...), but the shortest route modelled by this arc is computed according to one single criterion, generally travel time. Consequently, some alternative routes proposing a different compromise between the attributes of the arcs are discarded from the solution space. In this work, we propose to represent the road network with a multigraph, so that these alternative routes are considered, and to evaluate how it impacts on solution algorithms and solution values. A simple insertion algorithm is proposed and illustrated in the context of a on-demand transportation system developed in a French department. Computational experiments on academic and realistic data underline the potential cost savings brought by the multigraph model.

KEYWORDS: *vehicle routing, on-demand transportation, multigraph, shortest path problem with resource constraints, dial-a-ride problem.*

1 Introduction

The class of vehicle routing problems has drawn many researchers and industrial practitioners attention during the last decades. These problems involve the optimization of freight or passenger transportation activities. They are generally treated via the representation of the road network as a weighted complete graph, constructed as follows. The vertex set is the set of origin or destination points. Arcs represent shortest paths between pairs of vertices. Several attributes can be defined for one arc (travel time, travel cost ...), but the shortest path implied by this arc is computed according to one single criterion, generally travel time. Consequently, some alternative paths proposing a different compromise between these attributes are dismissed at once from the solution space, as illustrated in Figure 1. In the following, to avoid any ambiguity between the paths of the new graph (working graph in the figure) and the paths of the original road network, the latter are called road-paths.

This can be problematic in many situations. A typical example is provided by On-Demand Transportation (ODT) systems. In such systems, transportation plans need to be computed, to satisfy point-to-point transportation requests, according to some quality of service constraints and/or objectives. Though the road-path retained between two (origin or destination) customer locations is generally set as the min-time road-path, the driver or the shipper might prefer a cheaper itinerary if he gets time. If the customer pays according to the distance (which is generally not the case in ODT systems, but is true in taxis),



	road network	working graph
paths from 1 to 2	(D); (E); (A,C); (B,C)	(D); (A,C)
paths from 1 to 3	(A); (B)	(A)
paths from 3 to 2	(C)	(C)

Figure 1: Simple graph construction

avoiding fast but long-distance sections could also be of interest (for the customer). Note that computing the shortest path matrix according to distance instead of time could induce similar drawbacks, especially considering sections with heavy traffic. Section 4 will develop the example of an ODT system implemented in the Doubs Central area in France.

In this work, we propose to represent the road network with a multigraph, so that alternative routes are considered. Ideally, one arc will be added between two vertices for each Pareto optimal road-path according to arc attributes in the road network. Any good road-path would then be captured in the graph. Practically, one could prefer just to consider a reasonable set of arcs between two vertices.

At least two other situations would deserve to be further explored, but will be left as perspectives here. A first situation would be the case of a traveler having several transportation modes at his disposal (foot, metro, tramway, bus ...) and having to decide how to combine them to reach some destination. If transportation modes can be competitive for the same piece of trip, the multigraph representation appears to be well-suited as long as the schedule of facilities can be neglected (as it is often the case for a tramway or a metro for example, but not for a train). Several papers deal with multimodal transportation in the literature (Horn, 2002 and 2003; Bielli *et al.*, 2006). However, to the best of our knowledge, they all consider a single-request. Hence, the problem is to determine an optimal (or a set of optimal) trip from an origin to a destination in a multigraph, where arcs correspond to different transportation modes and nodes to interchange points. A second situation would be met by a touristic traveler. One might then have some clearly identified destination points and different possibilities (with different duration and touristic interests) of linking these points. Actually, having a multigraph representation makes sense as soon as several attributes are defined on arcs.

Although original, the use of a multigraph representation in the context of vehicle routing is not entirely new. A recent work by Baldacci *et al.* (2006) introduces a similar representation to solve the so-called *Multiple Disposal Facilities and Multiple Inventory Locations Rollon-Rolloff Vehicle Routing Problem*. The topic is to transport trailers between customers, disposal facilities and inventory locations. In this context, the multigraph dimension stems from the enumeration *a priori* of valid sequences of movements between customers. An exact solution method based on a *Set Partitioning* formulation and a sophisticated iterative bounding procedure are proposed.

In this paper, our first objective is to evaluate the tractability of the multigraph representation. We describe this representation in Section 2. Section 3 focuses on the new difficulties it implies. The paper concludes with the case of a practical ODT system in Section 4.

2 Multigraph representation

Let $G_0 = (V_0, A_0)$ be the graph induced by a road network. An arc of A_0 typically represents a link between two crossroads or a portion of road having consistent characteristics (slope, direction, sinuosity ...). G_0 has the advantage to offer a complete and precise description of the physical layout, but can reach a size detrimental to the efficient execution of routing optimization procedures. We consider

here that each arc $(i,j) \in A_0$ is characterized by R attributes ($R \geq 2$): $d_{ij}(1), \ldots, d_{ij}(R)$. Attributes can indifferently represent duration, distance, cost, interest, roughness, *etc.*

Let us assume that we are interested here in some vehicle routing problem. For sake of generality, we do not define it precisely. Let us name key-locations the set of all locations of G_0 playing a special role in the problem: vehicle depots, customer locations, origin and destination of transportation requests ... Let $V \subset V_0$ be the set of all key-locations. For $(i,j) \in V \times V$, let $\mathcal{P}_{ij}$ be the set of all Pareto optimal paths, in G_0, from i to j, considering the R criteria. We introduce the multigraph $G = (V, A)$. For each couple of vertices $(i,j) \in V \times V$ and each road-path $P_{ij}^e \in \mathcal{P}_{ij}$ ($1 \leq e \leq |\mathcal{P}_{ij}|$), we introduce an arc $(i,j)^e \in A$. Arc $(i,j)^e$ is then characterized by resource consumption levels $d_{ij}^e(1), \ldots, d_{ij}^e(R)$.

Note that sets $\mathcal{P}_{ij}$ are possibly of very large size. A first difficulty with the multigraph representation is to compute these sets. The problems to solve are *Multicriteria Shortest Path Problems*. A variety of algorithms based on dynamic programming (Warburton, 1987; Guerriero and Musmanno, 2001; Skriver and Andersen, 2000) are available in the literature (see also Ehrgott and Gandibleux, 2000, for an exhaustive survey). These methods are robust and can handle several types of objective functions like *min-sum* or *max-min*. This robustness is fundamental in our situation, where we might have to deal with various types of attributes. Theoretically, computing sets $\mathcal{P}_{ij}$ can be very time-consuming, especially when R is large. However, one can expect to have $R = 2$ or $R = 3$ in most practical cases. Also, attributes like time, distance and cost are generally closely correlated. This can drastically limit the number of Pareto optimal paths. Anyway, the time needed is not really the point of this paper, sets $\mathcal{P}_{ij}$ being computed only once.

3 Route optimization in a multigraph

Vehicle routing problems generally address three types of decisions:

- assignment decisions, allocating key-locations to vehicles;
- sequencing decisions, defining the visit order for each vehicle;
- scheduling decisions, determining a timetable for the visit of the assigned key-locations for each vehicle.

Once assignment and sequencing decisions are fixed, it is generally trivial to deduce timetables. With the multigraph representation, this property does not hold. Indeed, one has to determine which arc to use between two consecutive key-locations of the sequence. As shown below, this problem, we call *Fixed Sequence Arc Selection Problem* (FSASP), is NP-hard. Section 3.1 discusses the complexity of the FSASP and describes an efficient pseudo-polynomial solution algorithm based on dynamic programming. The remaining of the section investigates the impact of the multigraph representation on standard solution schemes. The particular case of the insertion operator, widely used in local search for vehicle routing, is developed in Section 3.3.

3.1 Fixed Sequence Arc Selection Problem

Let us define the FSASP more precisely. Let $G = (V, A)$ be a so-called linear multigraph[1] and consider $R + 1$ attributes as defined in Section 2. G is a fixed sequence $(i_0, \ldots, i_N)$ (see Figure 2). Attribute 0 corresponds to the objective function. An upper bound Q^r is defined for $1 \leq r \leq R$. The FSASP is to select a set of arcs $(i_0, i_1)^{e_1}, (i_1, i_2)^{e_2}, \ldots, (i_{N-1}, i_N)^{e_N}$ such that attribute 0 is minimized and upper bounds Q^r are satisfied for $1 \leq r \leq R$.

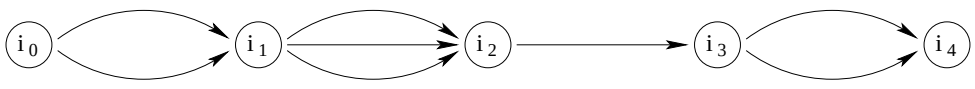


Figure 2: G is a linear multigraph

[1] We call a "linear multigraph" an acyclic graph such that a vertex has at most one predecessor and one successor.

For the sake of clarity, we only consider cumulative attributes here. Results can easily be extended to other types of attributes, like min-max ones. The problem can then be stated as follows:

(FSASP)

$$\min \sum_{n=1}^{N} \sum_{e=1}^{|\mathcal{P}_{i_{n-1}i_n}|} d^e_{i_{n-1}i_n}(0) y^e_{i_{n-1}i_n} \tag{1}$$

subject to

$$\sum_{n=1}^{N} \sum_{e=1}^{|\mathcal{P}_{i_{n-1}i_n}|} d^e_{i_{n-1}i_n}(r) y^e_{i_{n-1}i_n} \leq Q^r \qquad \forall r = 1, \ldots, R \tag{2}$$

$$\sum_{e=1}^{|\mathcal{P}_{i_{n-1}i_n}|} y^e_{i_{n-1}i_n} = 1 \qquad \forall n = 1, \ldots, N \tag{3}$$

$$y^e_{i_{n-1}i_n} \in \{0,1\} \quad \forall n = 1, \ldots, N, \forall e = 1, \ldots, \left|\mathcal{P}_{i_{n-1}i_n}\right| \tag{4}$$

Binary decision variables $y^e_{i_{n-1}i_n}$ represent the selection of arcs $(i_{n-1}i_n)^e$. Objective function (1) is to minimize the total cost (attribute 0) of selected arcs. Upper bounds on attributes are handled by constraints (2). Constraints (3) impose that exactly one arc is selected between two vertices.

With the above assumption, the FSASP exactly corresponds to the Multidimensional Multiple Choice Knapsack Problem (MMKP), an NP-hard generalization of the Knapsack Problem (Kellerer *et al.*, 2004). The MMKP can be described as follows. A set of N classes of objects are defined. Each object e in class n is characterized by R weights $d^e_n(r)$ $(1 \leq r \leq R)$ and a cost $d^e_n(0)$. A limit Q^r is defined for each dimension r $(1 \leq r \leq R)$. The problem is to select exactly one object per class, while satisfying limits Q^r and minimizing the total cost of the objects selected.

Few papers dealing directly with the MMKP are available. One can mention Hifi *et al.* (2006) and Akbar *et al.* (2004) that propose heuristic solution schemes. An exact method based on branch-and-bound is proposed in Sbihi (2003). We prefer to address the FSASP as a particular *Shortest Path Problem with Resource Constraints* (SPPRC) (Beasley and Christofides, 1989). Resources correspond to attributes; $d^e_{ij}(r)$ indicates the level of consumption of resource r when arc $(i,j)^e$ is traversed. The objective is to find a shortest path, connecting vertex i_0 to vertex i_N, while resource constraints are satisfied.

If we assume that resource consumptions are non-decreasing when labels are extended, the SPPRC can be solved with dynamic programming (Irnich and Desaulniers, 2005). One can expect most types of attributes to comply with this assumption. This approach is thus consistent with our objective of dealing with vehicle routing in general. The algorithm, first proposed by Desrochers and Soumis (1988), is an extension of the classical Bellman's algorithm. The principle is to associate with each possible partial path a label which contains the consumption level for each resource at the end of the partial path, and to extend these labels checking resource constraints until the best feasible paths are obtained. Dominance rules are used to compare partial paths arriving at a same location and to discard some of them. Unlike Bellman's algorithm, when no resources are considered, each vertex of the graph can maintain a large number of labels since the comparison of two labels takes into account their consumption level for each resource. The algorithm is not initially designed for the case of a multigraph, but remains valid in this context. One just has to consider every outgoing arc when extending labels. Algorithm 1 presents this algorithm. Figure 3 illustrates label extension and dominance rules on the example of a FSASP with 3 vertices and 2 resources. In this figure, among the 4 possible partial paths reaching i_2, one is dominated and one is unfeasible. The algorithm would thus only consider the two remaining labels to continue the sequence.

When searching for the optimal arc set to be selected in the sequence, the dynamic programming algorithm is applied on an acyclic graph of limited size (one can expect that in most cases a vehicle route visits a small number of vertices), which helps finding optimal solutions efficiently (Irnich and Desaulniers, 2004).

The complexity of the dynamic programming algorithm is $\mathcal{O}(NLR)$, where L is the maximal number of labels ending at a vertex, provided all resource extension functions have a constant complexity. When resource consumption values are integer, L is bounded by $\Pi_{r=1,\ldots R} Q^r$.

Data: G a sequence from 0 to N
Result: L_N
initialization : $L_0 := 0_R$;
for $i = 0$ *to* N **do**
  foreach *label* $l \in L_i$ **do**
    foreach *outgoing arc* a *from* i **do**
      $l' := l$ extension from i to j by a;
      $dominated := false$;
      foreach *label* l'' *in* L_j **do**
        if $l'(r)$ *dominates* $l''(r), \forall r = 0, \ldots, R$ **then**
          $L_j := L_j \setminus \{l''\}$;
        else
          if $l''(r)$ *dominates* $l'(r), \forall r = 0, \ldots, R$ **then**
            $dominated := true$;
            break;
          end
        end
      end
      if *dominated=false* **then**
        $L_j := L_j \cup \{l'\}$;
      end
    end
  end
end

Algorithm 1: Dynamic programming algorithm

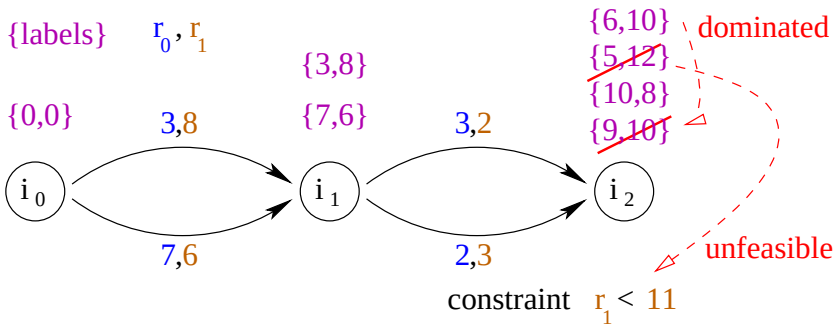


Figure 3: Dynamic programming algorithm for FSASP

Note that the SPPRC is very close to the *Multicriteria Shortest Path Problem* discussed in Section 2 for the initial construction of G, resources standing for criteria. As a matter of fact, we also use our dynamic programming algorithm at this stage of the resolution.

3.2 Impact on resolution algorithms

Local search algorithms basically consist in repeatedly considering an incumbent solution, exploring a set of neighbour solutions and selecting a new incumbent solution in this neighbourhood. In a simple descent algorithm, the best neighbour solution is selected at each iteration until it does not improve upon the incumbent solution. Several metaheuristic mechanisms can be added to avoid being trapped into local optima. The multigraph representation does not interfere with the local search scheme except for evaluating the feasibility and the value of the solutions explored, which is exactly the purpose of the FSASP.

However, one can be a little more clever than simply evaluating every neighbour solution using the dynamic programming algorithm of Section 3.1. A possibility would be to explore the whole neighborhood and find the best neighbour solution with one execution of the dynamic programming algorithm. This possibility is illustrated for the insertion operator in Section 3.3.

This latter operator is critical for inter-routes moves like *relocate* and *exchange*. *Cross-moves* which plug subsequences or intra-route neighbourhood operators (k-opt, Or-opt, ...) are quite different. Exploring the whole neighbourhood in one shot appears more tricky in these cases. One can however expect that the size of these neighbourhoods will be limited, especially when resources are very restrictive (*e.g.*, tight time windows).

With regards to exact methods, using a multigraph representation increases drastically the size of the solution space. Hence, one can conjecture that these methods would fail to solve instances of a size that they would be able to tackle with a simple graph representation. However, the basic principles of the methods would not be changed. Linear relaxation can still be computed and serve as a lower bound in a branch-and-bound method; one can expect most of the valid inequalities to remain true; column generation can be applied with a simple adaptation of the subproblem ... Concerning the branching scheme, usual branching decisions enforce or forbid the use of an arc. With the multigraph representation, this policy can be rather inefficient, as forbidding an arc is not as strong as in the simple graph case. One might rather prefer to enforce or forbid the successor of a vertex, *i.e.*, enforce or forbid the complete set of arcs between two vertices. It can be simply seen that in a fractional solution at least one vertex has two different successors and that adapting the classic branching rule this way will always be possible. Indeed, a fractional solution with a single successor for every vertex would be a solution with identical sequences using different subsets of arcs, which can easily be transformed into an integer solution.

3.3 Insertion in a sequence

The insertion operator consists in searching for the best insertion position of a given vertex s in the sequence. Let us introduce the acyclic multigraph $G_1 = (V_1, A_1)$ where V_1 is the vertex set of the sequence (including the depot(s)) plus one vertex for each possible insertion position for s; in the following we call virtual vertices these latter vertices; A_1 contains every arc respecting the sequence order. Figure 4 illustrates the construction of G_1. We adapt the Algorithm 1 mentioned in Section 3.1 by adding a resource R_s implying the insertion of exactly one virtual vertex in the sequence. Finding the best insertion position is then equivalent to finding the shortest resource constrained path in G_1.

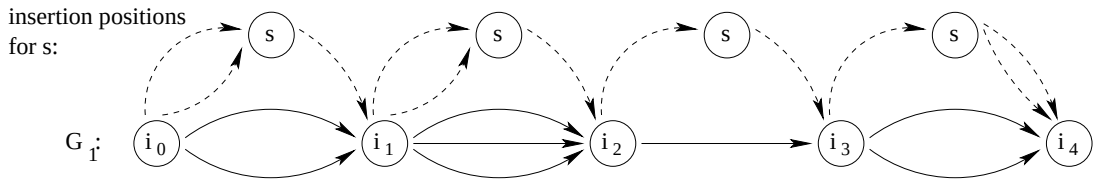


Figure 4: Insertion in a sequence – multigraph construction

The new resource R_s works as follows (see Table 1). R_s is initialized with a value 0. Ingoing arcs on virtual vertices consume 1 unit of resource, other arcs consume 0 unit. The value of the resource is constrained with an upper bound of value 1; labels can only be extended to the final depot when this upper limit is reached. The value of the resource is trivially non-decreasing when labels are extended.

arc	(i,j)	(i,s)	$(i,depot)$
consumption	0	1	0
violation	$-$	$\neq 0$	$\neq 1$

Table 1: Functioning of resource R_s

4 Example of an ODT system in the Doubs Central area

4.1 ODT description

The motivation of this study stems from a multidisciplinary research project dealing with on-demand transportation systems and their adequacy with new mobility practices (Josselin and Genre-Grandpierre, 2005). We focus here on the development of an ODT system in the Doubs Central area (France). An ODT system is a flexible transportation system intended to carry out transportation requests via a fleet of vehicles under feasibility and operational constraints. Contrary to a traditional public transportation system, routes are determined on a daily basis (or, at least, for a short time period), according to the requests. A key issue for such systems is to find operational solutions taking into account of the possibly contradictory objectives of the involved partners:

- for the Transportation Organizing Authorities: rationalize and make the service attractive;
- for the conveyors (local taxi companies in the Doubs Central case): maximize profits;
- for the possible subcontractors (hauliers): conquer new markets;
- for the passengers associations : improve quality of life and access to the facilities.

In the ODT system considered here, each user issues a request defined by a pick-up point (departure), a drop-off point (arrival), a number of passengers and a latest drop-off date that cannot be exceeded. An acceptable quality of service can be ensured by providing a guarantee on the maximal gap between the pick-up date and that latest drop-off date. As specified subsequently, these constraints can be expressed as time windows. The service is carried out by local taxi companies. Consequently, the fleet is heterogeneous, of fixed size and based in multiple depots. The cost for the authorities (paid to these taxi companies) is proportional to the distance traveled plus a fixed cost for every vehicle used. Our objective is to propose a transportation plan satisfying all requests and minimizing this cost. A second objective is then to minimize the time spent in vehicles by users.

The interest of considering alternative paths here is to have the possibility to propose less expensive paths (using *e.g.*, short but slow sections), with an equivalent quality of service. Also, this problem appears as a good test-bed case to evaluate the methodological and the practical impact of a multigraph representation.

ODT systems have raised the interest of many researchers for a long time. The underlying vehicle routing problem is generally identified as the *Dial-a-Ride Problem* (DARP). The DARP is a special case of *Pickup & Delivery Problem with Time Windows* (PDPTW), which consists in transporting goods from collection to delivery points. The specificity of the DARP pertains to the quality of service induced when carrying persons. Most of the work on the DARP is issued from realistic applications. Passenger flows are often important (up to thousands of people a day) even if most of the systems are reserved for specific categories like disabled people (Toth et Vigo, 1997; Dumas *et al.*, 1989). For both reasons, the DARP has so far been mostly investigated with heuristic approaches.

Insertion procedures are often applied to construct feasible solutions. They are fast, they are robust with regard to involved constraints and they are particularly adapted to dynamic situations. The insertion procedure of Jaw *et al.* (1986) is a reference in the context of the DARP. Authors deal with

individual maximal ride time and time window constraints, and instances with up to 2600 passengers and 20 vehicles. Requests are selected and inserted with a best insertion policy in the increasing order of their earliest pick-up date. Madsen *et al.* (1995) and Coslovitch *et al.* (2006) adapt this procedure to the dynamic case. Other insertion procedures are proposed by Toth and Vigo (1996) and Diana and Dessouky (2004). The obtained solutions are then generally improved using metaheuristics. Several evolved metaheuristics have recently been proposed, but these procedures still have difficulties to deal with complex side-constraints. Bent and Van Henteryck (forthcoming) and Ropke and Pisinger (2006) propose large neighbourhood search approaches, combined with simulated annealing. Cordeau and Laporte (2003a) and Melachrinoudisa *et al.* (2007) use tabu search. More recently, Xiang *et al.* (forthcoming) solve with local search a complex DARP including various customer and driver quality of service constraints.

Another direction for solving the DARP is to take advantage of the natural splitting of the problem into an assignment (*clustering*) and a sequencing (*routing*) subproblems – which can also contains a *scheduling* subproblem to determine service dates.

This structure can be used in decomposition schemes. Several branch-and-price approaches have been developed successfully (Dumas *et al.*, 1989 and 1991; Savelsbergh and Sol, 1998; Ropke, 2005). Another possibility is to solve sequentially the two subproblems. The *clustering* phase has been addressed efficiently with genetic algorithms (Rekiek *et al.*, 2006 and Jørgensen *et al.*, forthcoming) and simulated annealing (Li and Lim, 2001; Colorni *et al.*, 1996). The *routing* phase amounts to solving a single vehicle DARP. Sexton and Bodin (1985a,b) propose a Bender's decomposition where the slave problem is *scheduling*. Other authors generally use local or tabu search for heuristics and dynamyc programming for exact methods.

Finally, several efficient branch-and-cut methods have recently been developed. This kind of approach is known as the most successful for the TSP (Gutin and Punnen, 2002). Lu and Dessouky (2004) and Cordeau (forthcoming) solve small instances (less than 50 requests). Ropke *et al.* (forthcoming) tackle instances with hundreds of passengers, with new models and new valid inequalities.

The interested reader may find a more detailed state-of-the-art review on this subject in Cordeau and Laporte (2003) (a recent updated version is forthcoming). Desaulniers *et al.* (2002) and Crainic and Laporte (1998) present more general information on *Pickup & Delivery Problem* and other vehicle routing problems.

4.2 Problem Formulation

The problem is to serve a set $\mathcal{R}$ of requests with an heterogeneous fleet of K vehicles. A request r is defined by its pick-up point $r+$, its drop-off point $r-$, a positive number of passengers to transport l_{r^+}, a latest drop-off date B_{r^-} and the gap δ_r to respect between B_{r^-} and the actual pick-up date. Each vehicle k is characterized by its capacity C_k, a starting and an arrival depots o_k and m_k, and a fixed cost pc_k incurred when the vehicle is used. In the remaining of this section, we note $l_{r^-} = -l_{r^+}$ and $l_{o_k} = l_{m_k} = 0$, for each request r and vehicle k. Also, a pick-up or a drop-off point is called a service. Beside the characteristics defined above, a service i has a non-negative duration s_i.

Let $G = (V, A)$ be a directed multigraph. V includes two nodes per request, one for each service r^+ and r^-, plus two depot nodes per vehicle (starting and arrival depots). An arc $(i, j)^e \in A$ is a road-path linking node i to node j. A cost $d^e_{ij}(0)$, a load $d^e_{ij}(1) = l_j$ and a duration $d^e_{ij}(2)$ are associated with arc $(i, j)^e$. Indices $e = 0$ represent arcs corresponding to min-time road-paths.

The latest drop-off and maximal gap constraints can be expressed as time windows, as can be seen in equations (5)-(7):

$$B_{r^+} = B_{r^-} - d^0_{r^+r^-}(2) - s_{r^+} \quad (5)$$

$$A_{r^+} = B_{r^-} - \delta_r - s_{r^+} \quad (6)$$

$$A_{r^-} = A_{r^+} + s_{r^+} + d^0_{r^+r^-}(2) \quad (7)$$

where A_i is the earliest starting date for service i and B_i the latest starting date. This remark permits to replace the gap constraint (concerning two services) with time window constraints $[A_i, B_i]$ defined for every service i independently.

The part of a solution relative to a single vehicle is called *route*. It is called *sequence*, if the service dates are not fixed, *i.e.*, only the vehicle assignment and the order of realization of the *services* are known.

To construct the routing planning, one has to assign one vehicle per request, to sequence the services and to fix service starting dates T_i. The latter corresponds to the arc selection problem induced by the multigraph representation.

The problem can be modeled as follows. We introduce binary decision variables x^k_{ije}, with $x^k_{ije} = 1$ if arc $(i,j)^e$ is used by vehicle k, $x^k_{ije} = 0$ otherwise. Decision variables L_i indicate the number of passengers in the vehicle after service i is achieved. The model is then:

$$\min \text{Lex}\Big(\sum_{k\in K}\sum_{(i,j)^e\in A} x^k_{ije} d^e_{ij}(0) + \sum_{k\in K}\sum_{(o_k,i)^e\in A} pc_k x^k_{o_k ie}, \sum_{r\in\mathcal{R}}(B_{r^-} - T_{r^+})\Big) \tag{8}$$

subject to

$$\sum_{k\in K}\sum_{(r^+,j)^e\in A} x^k_{r^+je} = 1 \qquad \forall r \in \mathcal{R}, \tag{9}$$

$$\sum_{(o_k,j)^e\in A} x^k_{o_kje} = 1 \qquad \forall k \in K, \tag{10}$$

$$\sum_{(r^+,j)^e\in A} x^k_{r^+je} - \sum_{(j,r^-)^e\in A} x^k_{jr^-e} = 0 \qquad \forall k \in K, \forall r \in \mathcal{R}, \tag{11}$$

$$\sum_{(i,j)^e\in A} x^k_{ije} - \sum_{(j,i)^e\in A} x^k_{jie} = 0 \qquad \forall k \in K, \forall j \in V, \tag{12}$$

$$x^k_{ije}\left(T_i + s_i + d^e_{ij}(2) - T_j\right) \le 0 \quad \forall k \in K, (i,j)^e \in A, \tag{13}$$

$$A_i \le T_i \le B_i \qquad \forall k \in K, i \in V, \tag{14}$$

$$T_{r^+} + s_{r^+} + d^0_{r^+r^-}(2) \le T_{r^-} \qquad \forall k \in K, r \in \mathcal{R}, \tag{15}$$

$$x^k_{ije}\left(L_i + d^e_{ij}(1) - L_j\right) = 0 \quad \forall k \in K, (i,j)^e \in A, \tag{16}$$

$$L_{r^+}\sum_{(r^+,j)^e\in A} x^k_{r^+je} \le C_k \qquad \forall k \in K, r \in \mathcal{R}, \tag{17}$$

$$L_{r^-}\sum_{(i,r^-)^e\in A} x^k_{ir^-e} \le C_k + d^e_{ir^-}(1) \qquad \forall k \in K, r \in \mathcal{R}, \tag{18}$$

$$x^k_{ije} \in \{0,1\} \quad \forall k \in K, (i,j)^e \in A. \tag{19}$$

with $x^k_{o_kr^-} = x^k_{r^+m_k} = 0$; $x^k_{io_k} = 0$ for all $i \neq m_k$; $x^k_{m_ki} = 0$ for all $i \neq o_k$; $L_{o_k} = L_{m_k} = d^e_{m_ko_k}(0) = d^e_{m_ko_k}(1) = d^e_{im_k}(1) = d^e_{o_ki}(1) = 0$; $A_{o_k} = A_{m_k} = -\infty$ and $B_{o_k} = B_{m_k} = \infty$.

Constraints (9) enforce that exactly one vehicle passes through exactly one arc $(r^+,j)^e$ for each request r. This traduces the fact that each pick-up service request must be fulfilled by exactly one vehicle and exactly one alternative arc. Constraints (10) ensure that vehicle, leave their starting depot exactly once, possibly for a dummy run to their ending depot. Constraints (11) state that, for each request r, the drop-off service has to be performed by the same vehicle as the pick-up service. Constraints (12) are standard flow conservation constraints. Non-linear constraints (13) enforce the precedence constraint between the service date of two nodes visited consecutively by the same vehicle. Constraints (14) are the time window constraints (see equations (5)-(7)). Constraints (15) enforce for each request precedence between the pick-up and the drop-off services. Constraints (16) traduce the conservation of the load at each service. Constraints (17) and (18) ensure that a vehicle leaving a pick-up (drop-off) service has the required capacity.

4.3 Insertion heuristic

We propose a three-step algorithm illustrated in Figure 5.

1. The first step is a greedy insertion procedure which aims at constructing one arc sequence per vehicle satisfying all requests.
2. A descent method, based on removals and insertions, is then used to improve the set of sequences.
3. The arc sequences are scheduled (time-stamped) during the last step.

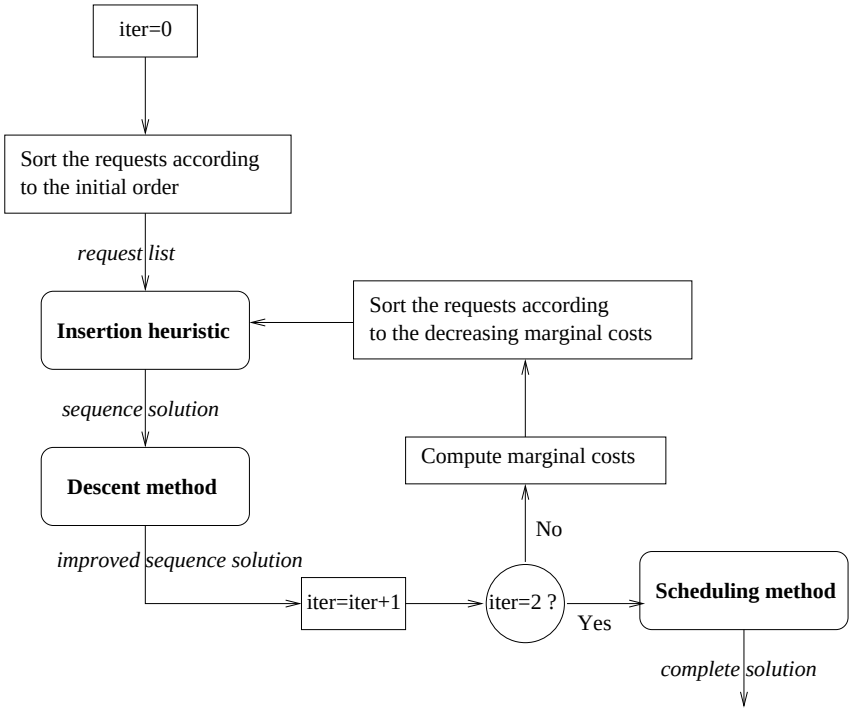


Figure 5: Our three-step approach

The main objective is tackled during steps 1 and 2 while the secondary objective is ignored. Step 3 optimizes the secondary objective while keeping the best value found for the main one. Before entering step 3, the method iterates between the insertion heuristic and the descent method as described in Section 4.3.1. Section 4.3.2 details the request insertion procedure as a special FSASP problem.

4.3.1 The three-step approach

Greedy insertion procedure Empty sequences are initially associated with each vehicle. The requests are then inserted one by one into the sequences, according to a predetermined order. The least-cost insertion is selected. Insertion is "greedy" in the sense that the relative order of the already inserted requests is preserved. However, the arc selection between successive stops is re-optimized. Insertion can indeed force to use faster but more expensive arcs. The determination of the optimal set of arcs is a NP-hard problem as a variant of the FSASP with 3 attributes, time, cost and capacity (see Section 3.1). The model and the method proposed to solve it are detailed in the following Subsection 4.3.2.

In order to improve the effectiveness of this phase, the algorithm is applied twice. Requests are treated in a predefined order. Two possibilities have been investigated: the earliest pick-up date and a random order. A marginal cost is then computed for each request. This cost is calculated by removing the request temporarily and by evaluating the corresponding profit, which impose a FSASP resolution on the new sequence. The order used for the second execution of the algorithm is then the decreasing order of the marginal costs, so that the most expensive requests are inserted first. The best of the two solutions obtained is preserved.

Descent method The descent method considers the set of sequences computed and try to move requests. Requests are removed and re-inserted, in the initial order of insertion which proved to be the most powerful. This operation is then randomly repeated until no more improvement is possible. A FSASP could need to be solved on the new sequence to re-optimize the arc selection. However in our case, removals always produce feasible sequences and this resolution is useless.

Scheduling (time-stamping) After the second step, sequences of arcs are chosen. The first hierarchical level of the objective function is then fixed. We denote S a such sequence. One can however optimize service dates according to the second hierarchical level, *i.e.*, minimize the sum of the gaps between latest drop-off dates (B_{r^-}) and pick-up dates (T_{r^+}). This amounts to minimize the following objective (20) since the latest drop-off times are constant:

$$\max \sum_{r \in S} T_{r^+} \tag{20}$$

Since arcs are fixed, we are in a classical context without multiple arcs to consider. Many contributions deal with this problem and different objective functions. Sexton and Bodin (1985a) minimize a weighted sum of drop-off date deviation and ride time. Desrosiers *et al.* (1995) generalize it to convex penalty cost functions and Ahuja *et al.* (2002) formulate and solve the scheduling problem with soft time windows as the *convex cost dual network flow problem*. In our special case, we describe a very simple adhoc procedure computing an optimal solution.

The recursion procedure described through formula (21), calculates the latest feasible service dates that characterize the *Latest Scheduling Solution*. $(i,j)^e$ denotes without any ambiguity the selected arc between i and j. By construction, we state easily that this scheduling is optimal for the secondary criterion.

$$T_{|S|} := B_{|S|};\ T_i := \min\left\{B_i, B_{i+1} - d^e_{i,i+1}(2) - s_i\right\} \quad \forall i = |S|-1, \ldots, 1 \tag{21}$$

Now, we consider service dates (T'_i) issued from the following rules and equations (22) and (23):

- A vehicle is allowed to wait at a stop only after a drop-off followed by a pick-up.
- As soon as the vehicle reaches a stop, the concerned passengers are dropped off.
- A vehicle leaves a stop immediately after the last pick-up.

$$\forall i \text{ a pick-up}, T'_i := T_i \tag{22}$$

$$\forall i \text{ a drop-off}, T'_i := T'_{i-1} + s_{i-1} + d^e_{i-1,i}(2) \tag{23}$$

Solutions respecting these operational constraints, called *Latest Pick-Up Earliest Drop-Off Solutions*, form a dominant set for the routing problem and the scheduling problem; pick-up dates are unchanged.

4.3.2 Request insertion in a sequence

This algorithm is an adaptation of the one described in Section 3.1. Let us consider the acyclic multigraph $G_1 = (V_1, A_1)$ where vertices are depots, pick-up and drop-off nodes of the sequence plus one vertex for r^+ and r^- at each insertion position (r is the request we have to insert). A_1 contains all arcs respecting the sequence order (Figure 6). Initial constraints or constraints induced by the sequence structure can immediately reduce the number of insertion positions.

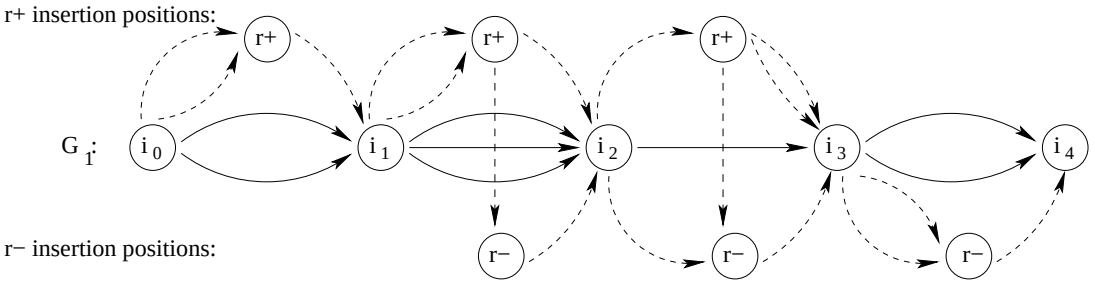


Figure 6: Request insertion in a sequence – multigraph construction

Finding the best insertion position for r is equivalent to find the min-cost path, satisfying all constraints on attributes (time and capacity) and visiting exactly one vertex representing r^+ and one representing r^- in this order. We model this problem as a SPPRC and solve it through dynamic programming

with the following particularities. The label lb associated with a partial path is defined with a level of consumption for each resource (time T^{lb} and load L^{lb}), a cost C^{lb}, and a final vertex i, plus the request resource R^{lb} for the new request. The rules of consumption and violation of resources T, L, C for extension of a label lb at a node i through arc $(i,j)^e$ are summarized in table 2. The rules for resource R are summarized in Table 3 depending on the destination vertex. The R^{lb} starting level at the depot is set to 0.

resource	value	constraint
$T^{lb'}$	$max\left\{T^{lb} + d^e_{ij}(2), A_j\right\}$	$\leq B_j$
$L^{lb'}$	$L^{lb} + d^e_{ij}(1)$	$\leq C$
$C^{lb'}$	$C^{lb} + d^e_{ij}(0)$	*to minimize*

Table 2: Extension functions for resources T, L and C

arc	(i,j)	(i,r^+)	(i,r^-)	$(i,depot)$
consumption	0	1	1	0
violation	–	$\neq 0$	$\neq 1$	$\neq 2$

Table 3: Extension function for resource R

The extension functions of these resources are trivially non-decreasing. Labels are generated traversing the sequence and considering all the outgoing arcs for every vertex. Labels violating constraints are deleted.

The dominance rule works as follow. lb_1 dominates lb_2 if equations (24) are valid.

$$C^{lb_1} \leq C^{lb_2}; T^{lb_1} \leq T^{lb_2}; L^{lb_1} \leq L^{lb_2}; R^{lb_1} \geq R^{lb_2} \tag{24}$$

The last condition on R^{lb_1} is valid by construction of G where each path (i,k,j) is dominated by one arc $(i,j)^e$. So each extension of lb_2 even coming through r^+ and r^- can be dominated by an extension of lb_1.

We discussed the complexity of the Algorithm 1 in Section 3.1. If we assume that service dates have integer values, we obtain for an insertion in G_1 a complexity of $\mathcal{O}(|V_1| \times C_k \times \max_{1\leq i\leq N}\{B_i\})$. We deduce a complexity of $\mathcal{O}(N^2 \times C_k \times \max_{1\leq i\leq N}\{B_i\})$ for the complete insertion procedure.

In addition, when the sum of the new request load and the maximal load in the sequence does not exceed the vehicle capacity, the load attribute can be ignored to intensify the dominance rule.

The algorithm efficiency is well improved by a constraint propagation procedure on time window constraints. After the new request insertion, time windows can be readjusted preserving all feasible solutions. The *Latest Scheduling Solution* (described in Section 4.3.1 in equation (21)) computed with min-time arcs ($e = 0$) give new bounds B_i and the *Earliest Scheduling Solution* – obtained by a symmetric construction – give new bounds A_i. This update is made traversing the sequence twice (one time in each direction) in $\mathcal{O}(|V_1|)$.

4.4 Results

Due to the many variants of DARP that can be considered, finding benchmark instances for these problems is not an easy task. Actually, no benchmark corresponds exactly to our situation, even with a simple graph representation. Cordeau generated two series (*'a'* and *'b'*) of random euclidean (on a [20x20] square) DARP instances, described in Cordeau (forthcoming). The instances are identified as follows: *xy_ z* where *x* is *a* or *b*, *y* is the number of available vehicles (the fleet is homogeneous at a single depot) and *z* is the number of transportation requests. In the first set (*'a'*), the vehicle capacity is 3 and there is only 1 passenger per request. In the second set (*'b'*), requests concern up to 6 passengers and the vehicle capacity is also 6. Each instance has a planning horizon, a common maximal individual ride time and a common time windows size. The pick-up time window is known for half of the requests, the *inbound* requests. The *outbound* requests impose a time window on the drop-off date. Since all requests have the same maximal individual ride time and the same time windows size, a drawback is that many passengers

are likely to be provided with a poor level of quality of service. There is also a small interest for using a multigraph with long maximal ride times, because min-cost arcs often form feasible routes. Thus, we derived from this benchmark a more time constrained one we denote as series *m_ 1* with an individual maximal ride time equal to 1.5 times the min-time path from pick-up to drop-off stops. We also generated 3 new instances (*'ab'* ones) by concatenation of the Cordeau instances with the same number of available vehicles. Two multigraph series were produced by adding arcs, N for a series we denote *m2_* (only $N/2$ for *m2_ ab*) and $6N$ for a series we denote *m6_* (only N for *m6_ ab*), and computing Pareto optimal paths. Added arcs are from 20% to 50% slower (according to attribute $d^e_{ij}(2)$) than initial arcs; they are cheaper (attribute $d^e_{ij}(0)$) in the same proportions. From 16 to 480 requests must be served by an unlimited homogeneous fleet. These instances are more deeply detailed in Annex A.

The results obtained with the algorithm presented in Section 4.3, are compared in Table 5 and 6 with those obtained by considering only the min-time paths (*m1_*) with a similar algorithm proposed in Garaix *et al.* (2005). Each instance is solved with the two orders indicated in Section 4.3.1. The Tables 5 and 6 show the best solution obtained. For each series we indicate the total distance (*'dist'*) of the solution, the computing time in seconds (*'cpu'*), the number of FSASP solved (*'FSASP'*) and the improvement due to the descent method (*'%Desc'*). We gain more than 7% in average with a descent method which takes 88% of the CPU. The descent can be stopped earlier with similar results. Using a multigraph reduces the distances obtained by 12.6% for *m2_* and $20,9\%$ for *m6_*. As expected, FSASP become harder and the average number of FSASP solved per second goes from 10944 for *m1_* to 3028 for *m2_* and to 1031 for *m6_*. These results show the strong impact of multigraph on both cost and CPU consumption. We programmed the algorithms in C++ and tests were ran on a 1.6 GHz Intel Centrino with 256 Mo RAM and a 300 seconds CPU time limit.

We also generated benchmark from geographical data of Doubs Central (using IGN[2] maps) and estimated flows of population. Three sets of 30 instances with 10, 30 and 90 requests were generated. The fleet is heterogeneous and corresponds to taxi companies. The maximal gap between the latest drop-off date and the actual pick-up date is two times longer than the min-time path (in the road network). The multigraph has from two to three arcs more than the simple graph. The variations between arcs connecting the same vertices can go up to 30% of gain in cost for 30% of additional travel time compared to the min-time path.

requests	graph	cost	vehicles	cpu (s)
10	SIMPLE	246	$3,0$	0
	MULTI	239	$3,2$	0
30	SIMPLE	578	$5,6$	1
	MULTI	559	$5,6$	2
90	SIMPLE	1368	$10,5$	30
	MULTI	1320	$10,6$	70

Table 4: Results on multigraphs and simple graphs on Doubs Central data

A software, jointly developed by the project team (UMR ESPACE, LIA) named TADOU© has been issued from this research and is currently operational in the Doubs Central area. Results obtained on the multigraph (MULTI) and on the simple graph (SIMPLE) are compared in Table 4. The number of requests and the algorithm used can respectively be found in the first and second columns. The three following columns indicate the average cost, number of vehicles used and computing times for the 30 instances of each benchmark. Six runs with different initial orders of requests are launched. The multigraph version obtains the least costly solutions on average (2.7%, 3.3% and 3.5% for the instances with 10, 30 and 90 requests respectively) with no more vehicles used. However, few results are better with the simple graph. We understand that the multigraph version leads first insertions to routes using long (according to time) arcs with min-cost. It prevents possibly better later insertions. Using a multigraph consume more computing time. Since latest improvements (in the descent method) are weak, reducing computing time has a weak impact on the quality of solutions. These tests were carried out with a 600MHz AMD Duron and 128 Mo RAM. The application is a C++ compiled kernel of a Microsoft Visual Basic application embedded in a database (Microsoft Access) environment.

[2]Institut Géographique National

instance	m1_				m2_				m6_			
	dist.	cpu	FSASP	%Desc	dist.	cpu	FSASP	%Desc	dist.	cpu	FSASP	%Desc
a2-16	322.21	0.0	382	5.4	290.13	0.1	327	3.8	249.82	0.4	622	5.0
a2-20	399.31	0.1	605	7.3	349.20	0.1	772	5.2	315.00	0.2	543	6.4
a2-24	454.78	0.1	1046	15.3	420.08	0.4	1105	7.2	350.33	2.8	1149	13.5
a3-18	343.12	0.0	578	1.4	300.56	0.1	482	1.2	272.02	0.1	414	0.1
a3-24	380.29	0.0	582	12.4	324.87	0.3	1025	10.2	296.12	0.6	696	3.8
a3-30	581.67	0.1	1056	10.8	510.28	0.3	1264	7.0	471.55	0.7	1072	4.9
a3-36	645.07	0.3	3199	10.8	572.61	1.3	2700	10.5	506.08	6.3	2893	11.2
a4-16	316.34	0.0	382	1.0	296.70	0.0	381	0.6	259.13	0.1	448	3.1
a4-24	460.88	0.1	698	4.3	413.58	0.2	808	3.8	375.48	0.4	880	4.4
a4-32	556.49	0.1	1017	4.2	494.99	0.3	1540	3.7	455.89	0.6	1027	2.7
a4-40	706.97	0.1	1528	6.7	624.67	0.5	1633	7.6	560.85	4.1	2932	4.9
a4-48	820.12	0.3	1658	3.2	726.88	1.5	2613	5.8	653.67	3.7	2112	6.8
a5-40	658.01	0.2	1489	6.0	581.67	1.4	2842	9.7	524.47	3.6	3761	8.2
a5-50	830.16	0.4	2564	10.7	719.06	3.4	5029	7.6	659.32	5.4	2704	7.8
a5-60	1024.41	0.6	4089	8.1	878.08	3.7	3402	8.3	793.75	24.4	6270	6.8
a6-48	786.68	0.2	2138	5.5	690.68	1.1	3094	7.6	627.29	4.3	2301	8.4
a6-60	1073.02	0.7	5213	11.6	928.19	4.0	4850	5.7	846.78	10.1	3980	8.8
a6-72	1181.59	1.7	10862	12.3	1033.10	4.5	3520	7.9	934.72	20.7	4346	5.7
a7-56	938.45	0.5	3966	11.4	829.02	2.9	3985	12.0	752.75	8.9	4186	9.3
a7-70	1190.92	0.4	2799	2.8	1008.33	5.7	7296	9.9	923.28	10.1	4910	7.2
a7-84	1390.99	0.7	3676	5.4	1182.09	6.7	4533	5.7	1064.49	61.2	14097	11.1
a8-64	992.04	0.5	4267	10.7	846.13	3.6	4590	5.5	804.28	11.9	3678	10.1
a8-80	1198.65	0.9	5391	5.4	1029.10	10.1	7996	14.9	936.76	37.9	10784	8.7
a8-96	1566.28	1.6	8988	13.6	1367.76	15.0	10943	6.1	1238.40	93.7	12461	7.8

Table 5: Results on multigraphs and simple graphs

instance	m1_				m2_				m6_			
	dist.	cpu	FSASP	%Desc	dist.	cpu	FSASP	%Desc	dist.	cpu	FSASP	%Desc
b2-16	309.03	0.0	494	10.3	277.56	0.1	362	6.9	251.98	0.2	400	11.6
b2-20	330.38	0.0	563	14.3	287.05	0.1	486	4.8	257.33	0.6	712	8.2
b2-24	453.40	0.1	575	6.4	379.06	0.3	732	4.9	339.62	0.6	598	4.1
b3-18	330.15	0.0	794	12.7	288.72	0.1	824	3.6	253.05	0.1	386	1.4
b3-24	390.40	0.1	1298	11.6	350.39	0.2	1068	8.9	322.63	0.9	1781	10.5
b3-30	529.81	0.2	1919	11.6	457.65	0.6	1953	7.7	427.00	1.4	1342	6.6
b3-36	632.56	0.2	1469	5.0	560.49	0.7	1553	4.2	483.75	1.5	1325	2.9
b4-16	294.53	0.0	438	7.8	268.19	0.1	358	8.0	232.09	0.2	491	2.5
b4-24	394.35	0.1	714	9.5	347.86	0.2	1257	8.3	320.78	0.4	1267	11.8
b4-32	556.56	0.1	1167	12.1	476.79	0.6	1456	9.1	443.53	1.2	1178	4.1
b4-40	701.91	0.2	1560	4.2	595.41	1.3	2844	7.3	537.64	2.0	1262	3.7
b4-48	754.28	0.4	3441	13.8	680.56	1.6	2624	11.2	595.11	5.1	2218	12.5
b5-40	723.57	0.3	2871	10.9	617.77	0.9	1751	1.1	574.55	3.8	1862	2.0
b5-50	882.66	0.4	3245	10.5	766.80	2.1	5415	11.5	699.34	5.2	2976	7.3
b5-60	1051.23	0.6	4330	11.2	914.91	2.6	3425	4.8	826.34	9.3	3574	2.8
b6-48	803.72	0.2	2018	5.2	702.20	0.6	1778	4.0	635.06	6.5	3567	6.0
b6-60	1012.86	1.0	5866	8.6	849.86	2.4	2652	6.3	771.03	23.2	7136	13.6
b6-72	1147.85	1.3	8397	9.0	961.72	7.4	7377	11.0	891.70	18.1	4834	9.1
b7-56	978.34	0.4	3574	7.8	865.44	2.1	3069	4.7	787.38	11.5	4163	6.6
b7-70	1072.24	1.2	7868	9.2	906.64	3.6	4428	7.9	842.89	21.8	8554	10.6
b7-84	1446.52	1.0	5392	3.1	1210.26	10.7	6295	3.7	1112.64	36.6	6457	5.9
b8-64	1016.06	0.5	3728	5.2	892.52	2.8	4018	5.6	807.23	9.4	4245	9.0
b8-80	1234.54	1.8	11057	10.6	1121.77	5.4	6754	9.6	1016.28	19.5	5654	5.8
b8-96	1456.51	1.0	5516	2.2	1255.38	9.9	9944	3.8	1168.77	52.9	10835	12.4
ab2-120	2007.00	4.4	17316	6.4	1786.47	11.5	12635	9.9	1724.89	38.0	20494	95.2
ab6-360	5269.45	54.2	115370	13.0	4685.19	166.7	66479	5.3	4511.40	300.7	63236	85.1
ab8-480	6604.10	65.1	124403	5.3	5532.37	300.4	57011	3.2	5432.71	300.2	38709	26.3

Table 6: Results on multigraphs and simple graphs

5 Conclusion

In this article, we investigate the interest and the tractability of the use of a multigraph representation for solving vehicle routing problems where arcs with several attributes characterize alternatives in the road network.

In a first step, we make clear that this representation makes the problem harder even when the vehicle assignment and sequencing decisions are fixed, *i.e.*, when the problem is reduced to arc selection and service scheduling. After having shown that this scheduling subproblem is NP-hard, we propose to address it with a dynamic programming algorithm, based on a SPPRC modeling.

We then discuss how classical solution schemes, either based on local search (heuristics and metaheuristics) or enumeration (exact algorithms), can handle the multigraph representation. Though these points are only sketched out here, a special emphasis is made on the insertion operator.

An algorithm is derived in the case of an ODT system developed in the Doubs Central area in France. The insertion operator is specified for a request made of a pick-up and a drop-off service and a heuristic based on a greedy constructive algorithm and a descent method is presented. The computational study shows the efficiency and effectiveness of our algorithm for a set of benchmark instances issued from real data. These results permit to conclude positively concerning the tractability of the multigraph representation.

This work offers at least two important perspectives. A first one concerns the use of this representation in other contexts: multimodal networks, scenic route planning or road traffic congestion modeling have been underlined.

A second perspective is to investigate more deeply the adaptation of different algorithms to the multigraph representation. The issue is rather different for heuristic or exact algorithms. A first step is done concerning local search type algorithms. The solution scheme proposed to evaluate in a single run every neighbour solution for the insertion operator is indeed a good starting point for proposing equivalent algorithms for other types of operators. In any case, this paper shows that any type of operator can be used, if one accepts that the evaluation of a neighbour solution involves a SPPRC resolution.

Concerning exact methods, column generation appears as a very natural tool to cope with the multigraph representation. One can see at least two reasons for that. First, column generation proved to be very efficient in many routing situations. Second, a multigraph representation mainly impacts on the column generation subproblem, which happens to be a SPPRC, similar to ours, and which can directly integrates the multigraph dimension.

References

R.K. Ahuja, D.S. Hochbaum, and J.B. Orlin. Solving the convex cost integer dual network flow problem. *Management Science*, 49(7):950–964, 2003.

M.M. Akbar, M.S. Rahman, M. Kaykobad, E.G. Manning, and G.C. Shoja. Solving the multidimensional multiple-choice knapsack problem by constructing convex hulls. *Computers & Operations Research*, 33:1259–1273, 2004.

R. Baldacci, L. Bodin, and A. Mingozzi. The multiple disposal facilities and multiple inventory locations rollon-rolloff vehicle routing problem. *Computers & Operations Research*, 33(9):2667–2702, 2006.

J.E. Beasley and N. Christofides. An algorithm for the resource constrained shortest path problem. *Networks*, 19:379–394, 1989.

R. Bent and P. Van Henteryck. A two-stage hybrid algorithm for pickup and delivery vehicle routing problems with time windows. *Computers & Operations Research*, Forthcoming.

M. Bielli, A. Boulmakoul, and H. Mouncif. Object modeling and path computation for multimodal travel systems. *European Journal of Operational Research*, 175:1705–1730, 2006.

A. Colorni, M. Dorigo, F. Maffioli, V. Maniezzo, G. Righini, and M. Trubian. Heuristics from nature for hard combinatorial optimization problems. *International Transactions in Operational Research*, 3:1–21, 1996.

J.-F. Cordeau. A branch-and-cut algorithm for the dial-a-ride problem. *Operations Research*, Forthcoming.

J.-F. Cordeau and G. Laporte. The dial-a-ride problem (DARP): Variants, modeling issues and algorithms. *4OR*, 1:89–101, 2003.

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

J.-F. Cordeau and G. Laporte. The dial-a-ride problem : models and algorithms. *4OR*, Forthcoming.

L. Coslovich, R. Pesenti, and W. Ukovich. A two-phase insertion technique of unexpected customers for a dynamic dial-a-ride problem. *European Journal of Operational Research*, 175:1605–1615, 2006.

T.G. Crainic and G. Laporte. *Fleet Management and Logistics*. Kluwer, Boston, USA, 1998.

G. Desaulniers, J. Desrosiers, A. Erdmann, M.M. Solomon, and F. Soumis. VRP with pickup and delivery. In *The Vehicle Routing Problem*, pages 225–242. P. Toth and D. Vigo, Philadelphia, 2002.

M. Desrochers and F. Soumis. A generalized permanent labelling algorithm for the shortest path problem with time windows. *Transportation Science*, 26(3):191–212, 1988.

J. Desrosiers, Y. Dumas, M.M. Solomon, and F. Soumis. Time constrained routing and scheduling. In M.O. Ball, T.L. Magnanti, C.L. Monna, and G.I. Nemhauser, editors, *Network Routing*, Handbooks in Operations Research and Management Science, pages 35–139. Amsterdam, North-Holland, 1995.

Y. Dumas, J. Desrosiers, and F. Soumis. Large scale multi-vehicle dial-a-ride systems. *GERAD, Ecole des Hautes Etudes Commerciales, Montréal*, G-89-30, 1989.

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

M. Ehrgott and X. Gandibleux. A survey and annotated bibliography of multiobjective combinatorial optimization. *OR Spectrum*, 22:425–460, 2000.

T. Garaix, D. Josselin, D. Feillet, C. Artigues, and E. Castex. Transport à la demande points à points en zone peu dense. Proposition d'une méthode d'optimisation de tournées. *Cybergeo European Journal of Geography*, pages on–line, 12 p., 2005. Selection of acts from the SAGEO'2005 conference.

F. Guerriero and R. Musmanno. Label correcting methods to solve multicriteria shortest path problems. *Journal of optimization theory and applications*, 111(3):589–613, 2001.

G. Gutin and A.P. Punnen. *The Traveling Salesman Problem and its variations*. Kluwer Academic Publishers, Dordrecht, 2002.

M. Hifi, M. Michrafy, and A. Sbihi. A reactive local search-based algorithm for the multiple-choice multi-dimensional knapsack problem. *Computational Optimization and Applications*, 33:271–285, 2006.

M.E.T. Horn. Multi-modal and demand-responsive passenger transport systems: a modelling framework with embedded control systems. *Transportation Research A*, 36:167–188, 2002.

M.E.T. Horn. An extended model and procedural framework for planning multi-modal passenger journeys. *Transportation Research B*, 37:641–660, 2003.

S. Irnich and G. Desaulniers. Shortest path problems with resource constraints. In G. Desaulniers, J. Desrosiers, and M.M. Solomon, editors, *Column generation*, pages 33–66. Springer, 2005.

J.J. Jaw, A.R. Odoni, H.N. Psaraftis, and N.H.M. Wilson. A heuristic algorithm for the multi-vehicle many-to-many advance request dial-a-ride problem. *Transportation Research B*, 20B:243–257, 1986.

H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack problems*. Springer, 2004.

H. Li and A. Lim. A metaheuristic for the pickup and delivery problem with time windows. In *ICTAI 2001, Dallas, USA*, pages 160–170, 2001.

Q. Lu and M.M. Dessouky. An exact algorithm for the multiple vehicle pickup and delivery problem. *Transportation Science*, 38:503–514, 2004.

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

E. Melachrinoudisa, A.B. Ilhan, and H. Min. A dial-a-ride problem for client transportation in a health-care organization. *Computers & Operations Research*, 34:742–759, 2007.

S. Ropke. *Heuristic and exact algorithms for vehicle routing problems*. PhD thesis, University of Copenhagen (DIKU), 2005.

S. Ropke, J.-F. Cordeau, and G. Laporte. Models and branch-and-cut algorithm for pick-up and delivery problems with time windows. *Networks*, Forthcoming.

S. Ropke and D. Pisinger. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40(4):455–472, 2006.

M.W.P. Savelsbergh and M. Sol. DRIVE : Dynamic routing of independant vehicles. *Operations Research*, 46:474–490, 1998.

A. Sbihi. *Les méthodes hybrides en optimisation combinatoire : algorithmes exacts et heuristiques*. Phd thesis, Université Paris 1, 2003.

T. Sexton and L.D. Bodin. Optimizing single vehicle many-to-many operations with desired delivery times : I. scheduling. *Transportation Science*, 19:378–410, 1985.

T. Sexton and L.D. Bodin. Optimizing single vehicle many-to-many operations with desired delivery times : II. routing. *Transportation Science*, 19:411–435, 1985.

A.J.V. Skriver and K.A. Andersen. A label correcting approach for solving bicriterion shortest-path problems. *Computers & Operations Research*, 27:507–524, 2000.

P. Toth and D. Vigo. Fast local search algorithms for the handicapped persons transportation problem. In *Meta-heuristics: Theory and applications.*, pages 677–690. I.H. Osman & J.P. Kelly, Boston, 1996.

P. Toth and D. Vigo. Heuristic algorithms for the handicapped persons transportation problem. *Transportation Science*, 31:60–71, 1997.

A. Warburton. Approximation of pareto optima in multiple-objective, shortest-path problems. *Operations Research*, 35:70–79, 1987.

Z. Xiang, C. Chu, and H. Chen. A fast heuristic for solving a large-scale static dial-a-ride problem under complex constraints. *European Journal of Operational Research*, Forthcoming.

A Benchmark characteristics

Table 7 summarizes the main characteristics of the generated instances. H denotes the planning horizon length. Column TW indicates the average of time windows tightness ($B_i - A_i$) and ΔTW the average gap between pick-up and drop-off time windows ($B_{r^-} - A_{r^+}/d^0_{r^+r^-}(2)$), in the original Cordeau's instances. $TW2$ and $\Delta TW2$ indicate the same values for the new generated instances. The four last columns show the density of the initial multigraph G_0 (the road network) and of the reduced multigraph G (the services connections), for *m1_* , *m2_* and *m6_* series. The graph density is computed as $d(G = (V, E)) = |E|/|V|^2$. The initial graph represents the road network between stops and depot. The reduced graph is the graph of services and non-dominated paths between services stops. An arc in G_0 generates one in G, if it does not violate capacity or time window constraints. An arc from a service of the request i (i^+ or i^-) to a service of the request j (j^+ or j^-) has to be used in a valid sequence arranging requests i and j. Six such sequences exist: $(i^+ \to i^- \to j^+ \to j^-)$; $(i^+ \to j^+ \to j^- \to i^-)$; $(i^+ \to j^+ \to i^- \to j^-)$; $(j^+ \to j^- \to i^+ \to i^-)$; $(j^+ \to i^+ \to i^- \to j^-)$; $(j^+ \to i^+ \to j^- \to i^-)$.

instance	TW	ΔTW	$TW2$	$\Delta TW2$	$d_{m2}(G_0)$	$d_{m6}(G_0)$	$d_{m1}(G)$	$d_{m2}(G)$	$d_{m6}(G)$
a2-16	21.1	3.8	3.0	1.5	2.3	3.8	0.14	0.30	0.51
a2-20	21.9	4.4	2.4	1.5	2.2	3.4	0.14	0.31	0.46
a2-24	21.3	3.9	2.9	1.5	2.3	4.6	0.14	0.32	0.61
a3-18	21.7	4.2	2.4	1.5	2.4	3.6	0.13	0.27	0.37
a3-24	22.1	4.6	2.2	1.5	2.2	3.9	0.13	0.28	0.51
a3-30	21.5	4.1	2.7	1.5	2.6	4.0	0.13	0.31	0.47
a3-36	21.3	4.0	2.8	1.5	2.7	4.5	0.13	0.35	0.55
a4-16	21.4	4.0	2.8	1.5	2.3	3.8	0.13	0.25	0.41
a4-24	21.6	4.1	2.6	1.5	2.4	3.6	0.13	0.30	0.44
a4-32	21.6	4.1	2.6	1.5	2.5	4.0	0.12	0.27	0.43
a4-40	21.8	4.3	2.4	1.5	2.6	4.2	0.12	0.34	0.54
a4-48	21.9	4.5	2.3	1.5	2.7	4.5	0.13	0.30	0.53
a5-40	22.1	4.6	2.1	1.5	2.7	4.5	0.12	0.33	0.54
a5-50	21.6	4.2	2.5	1.5	2.8	4.5	0.13	0.34	0.54
a5-60	21.5	4.1	2.6	1.5	3.0	5.2	0.12	0.38	0.65
a6-48	21.9	4.4	2.2	1.5	2.8	5.1	0.12	0.34	0.58
a6-60	21.3	4.0	2.9	1.5	3.0	4.9	0.12	0.37	0.59
a6-72	21.8	4.3	2.5	1.5	3.2	5.2	0.12	0.40	0.63
a7-56	21.7	4.2	2.5	1.5	3.2	4.8	0.12	0.38	0.57
a7-70	21.7	4.2	2.5	1.5	3.2	5.1	0.12	0.37	0.60
a7-84	21.5	4.1	2.7	1.5	3.3	5.4	0.12	0.40	0.65
a8-64	22.0	4.5	2.3	1.5	3.2	5.3	0.12	0.39	0.63
a8-80	22.1	4.6	2.2	1.5	3.4	5.2	0.12	0.38	0.60
a8-96	21.4	4.1	2.7	1.5	3.5	5.8	0.12	0.43	0.67
b2-16	29.0	5.5	2.7	1.5	2.1	3.5	0.14	0.29	0.46
b2-20	28.6	6.3	1.5	1.6	2.2	3.6	0.13	0.34	0.55
b2-24	28.2	5.3	2.6	1.5	2.6	3.9	0.14	0.37	0.54
b3-18	28.4	5.6	1.9	1.5	2.1	3.4	0.13	0.26	0.42
b3-24	29.1	6.3	1.9	1.6	2.4	3.7	0.13	0.28	0.46
b3-30	28.2	5.3	2.4	1.5	2.4	4.1	0.13	0.32	0.52
b3-36	28.5	5.6	2.1	1.5	2.6	4.1	0.13	0.31	0.50
b4-16	27.1	5.0	2.0	1.5	2.3	3.8	0.13	0.29	0.48
b4-24	29.3	6.6	1.7	1.6	2.1	3.4	0.13	0.26	0.38
b4-32	28.6	5.9	2.1	1.6	2.5	4.3	0.13	0.33	0.54
b4-40	28.3	5.7	2.0	1.6	2.8	4.5	0.13	0.34	0.53
b4-48	29.5	6.3	2.2	1.6	2.9	4.9	0.13	0.34	0.54
b5-40	28.6	5.3	2.9	1.5	2.8	4.6	0.12	0.35	0.56
b5-50	28.3	5.4	2.6	1.6	2.7	4.5	0.12	0.31	0.51
b5-60	28.9	5.5	2.6	1.5	3.1	5.3	0.12	0.37	0.62
b6-48	28.2	5.5	2.0	1.5	2.9	5.1	0.12	0.33	0.60
b6-60	28.3	5.4	2.2	1.5	3.1	5.2	0.12	0.37	0.63
b6-72	28.8	5.8	2.3	1.6	3.3	5.3	0.12	0.40	0.63
b7-56	28.5	5.5	2.3	1.5	3.1	5.3	0.12	0.35	0.61
b7-70	29.1	6.5	1.9	1.6	3.3	5.1	0.12	0.38	0.62
b7-84	28.5	5.3	2.8	1.5	3.5	5.8	0.12	0.43	0.69
b8-64	29.1	6.2	1.9	1.6	3.2	5.3	0.12	0.35	0.58
b8-80	28.7	6.0	2.0	1.6	3.2	5.4	0.12	0.38	0.63
b8-96	29.2	6.0	2.3	1.6	3.1	4.9	0.12	0.37	0.59
ab2-120	-	-	2.5	1.5	2.9	3.7	0.12	0.34	0.42
ab6-360	-	-	2.4	1.5	3.1	4.1	0.12	0.37	0.48
ab8-480	-	-	2.3	1.5	4.8	6.1	0.12	0.55	0.71

Table 7: Instance characteristics