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 [a]

Thierry Garaix [a,c] Christian Artigues [b] Dominique Feillet [a]
Didier Josselin [c]

[a] *Université d'Avignon et des Pays de Vaucluse,*
Laboratoire Informatique d'Avignon (EA 931), F-84911 Avignon, France.

[b] *Université de Toulouse,*
LAAS-CNRS, 7 Avenue du Colonel Roche 31077 Toulouse, France.

[c] *Université d'Avignon et des Pays de Vaucluse,*
UMR ESPACE 6012 CNRS, F-84911 Avignon, France.

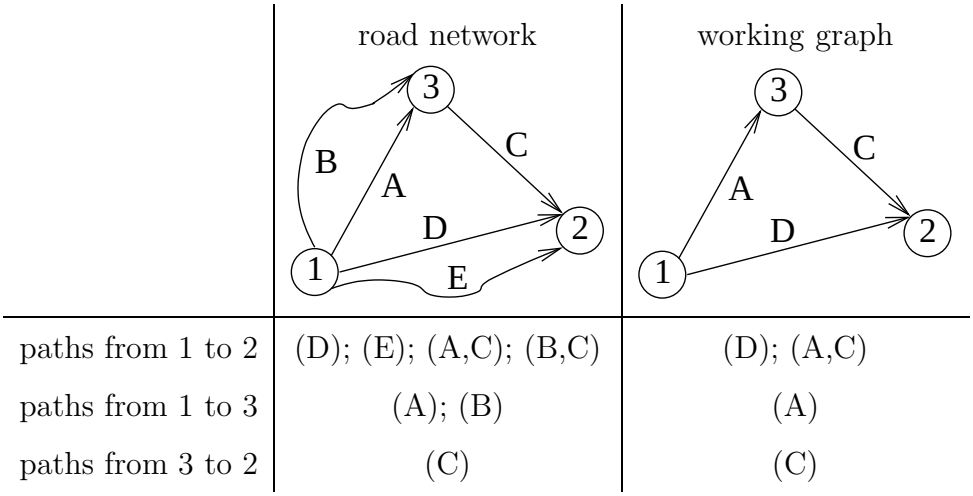
Abstract

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 modeled by this arc is computed according to a 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. We propose to consider these alternative routes and to evaluate their impact on solution algorithms and solution values through a multigraph representation of the road network. We point out the difficulties brought by this representation for general vehicle routing problems, which drives us to introduce the so-called fixed sequence arc selection problem (FSASP). We propose a dynamic programming solution method for this problem. In the context of an on-demand transportation (ODT) problem, we then propose a simple insertion algorithm based on iterative FSASP solving and a branch-and-price exact method. Computational experiments on modified instances from the literature and on realistic data issued from an ODT system in the French Doubs Central area underline the cost savings brought by the proposed methods using the multigraph model.

Key words: vehicle routing, multigraph, shortest path problem with resource constraints, dynamic programming, on-demand transportation, 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 a single criterion, generally travel time. Consequently, some alternative paths proposing a different compromise between these attributes are discarded from the solution space, as illustrated in Figure 1. In the remainder of this paper, we call road-



	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)

Fig. 1. Simple graph construction

paths, the paths of the original road network so as to distinguish them from the paths relative to the new graph (working graph in the figure).

Not considering alternative paths can be disadvantageous 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 in case time is non-critical. If the customer pays according to the distance (which is generally not the case in ODT systems, but is true in taxis), 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

Email address: thierry.garaix@univ-avignon.fr (Thierry Garaix).

drawbacks, for example 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 by a multigraph, so that alternative routes are considered. Ideally, this approach would define one arc 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. In practice, 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 is proposed.

In this paper, our first objective is to evaluate the tractability and the interest of the multigraph representation. Our second objective is to propose an efficient vehicle routing solution scheme for an actual ODT system. We describe the multigraph representation in Section 2. Section 3 focuses on the new difficulties it implies for general vehicle routing problem solving and proposes a dynamic programming solution method for the underlying fixed sequence arc

selection problem. In the context of a practical ODT system, Section 4 presents an insertion heuristic and an exact branch-and-price method and evaluates their results on modified instances from the literature and on real-life data issued from an ODT system in the French Doubs Central area. Concluding remarks are drawn in Section 5.

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 $R + 1$ attributes ($R \geq 1$): $d_{ij}(0), \ldots, d_{ij}(R)$. 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 at this point, while an application to an on-demand transportation system is described in Section 4. 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 $R + 1$ 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 $d_{ij}^e(0), \ldots, d_{ij}^e(R)$.

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 a survey). These methods are robust and can handle several types of objective functions like *min-sum* or *max-min*. This robustness is interesting 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 $R = 1$ or $R = 2$ in most practical cases. Also, attributes like time, distance and cost are generally closely correlated, which can drastically limit the number of Pareto optimal paths. Finally, in case of ODT systems, sets $\mathcal{P}_{ij}$ are going to be computed only once, prior to the optimization. In the following, we assume that the

construction of these sets could be carried out using any kind of method (*e.g.*, exact solution approach, heuristic, decision support system ...).

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 key location 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 for a standard min-sum criterion. 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, which 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. Section 3.2 shortly discusses the impact of the multigraph representation on standard solution schemes. This discussion will be continued further and illustrated with the case of an ODT system in Section 4.

3.1 Fixed Sequence Arc Selection Problem

Let us call linear multigraph a connected acyclic directed graph such that a vertex has at most one predecessor and one successor. A linear multigraph represents a fixed sequence $(i_0, \ldots, i_N)$ (see Figure 2). Let $G_{FS} = (V_{FS}, A_{FS})$ be a linear multigraph obtained from G, with $V_{FS} = \{i_0, \ldots, i_N\} \subset V$ and with arcs $(i_{n-1}, i_n)^e$ for $1 \leq n \leq N$ and $1 \leq e \leq \left|\mathcal{P}_{i_{n-1}i_n}\right|$. Let consider $R+1$ attributes $0, \ldots, R$ as defined in Section 2, with attribute 0 corresponding to the arc cost involved in the objective function. An upper bound Q^r is defined for $1 \leq r \leq R$. The FSASP amounts 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 $\sum_{n=1}^{N} d^{e_n}_{i_{n-1}i_n}(0)$ is minimized and upper bounds Q^r are satisfied for $1 \leq r \leq R$.

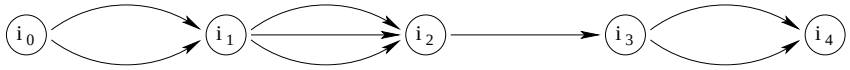


Fig. 2. Structure of a linear multigraph

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 the following integer linear program:

(FSASP)

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

subject to

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

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

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

Binary decision variables $y^e_{i_{n-1}i_n}$ represent the selection of arcs $(i_{n-1}, i_n)^e$. Objective function (1) (to be minimized) is 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 consecutive vertices.

We remark the FSASP corresponds exactly to the *Multidimensional Multiple Choice Knapsack Problem* (MMKP), an NP-hard generalization of the *Knapsack Problem* (Kellerer *et al.*, 2004), which proves that the FSASP is NP-hard. 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). Based on previous results on standard vehicle routing problems, we propose 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 only non-decreasing functions compute the cumulative resource consumptions, the SPPRC can be solved with dynamic programming

(Irnich and Desaulniers, 2005). One can expect most types of attributes (resources) to comply with this assumption. This approach is thus consistent with our objective of dealing with vehicle routing in general. The dynamic programming algorithm, first proposed by Desrochers and Soumis (1988) for the *Shortest Path Problem with Time Windows*, is an extension of the classical Bellman's labelling algorithm. We propose below a simple variant for the FSASP described by Algorithm 1.

The algorithm maintains a set of labels, each one corresponding to a partial path issued from i_0. More precisely , the algorithm runs in N iterations, each iteration $q \in \{0, \ldots, N-1\}$ being associated with a *dominant* label set L_q such that each label $l \in L_q$ represents a path from i_0 to i_q. Each label set L_q is dominant in the sense that either there exists an optimal path l^* from i_0 to i_N having a subpath $l \in L_q$ or there is no solution.

Each label l contains the cumulative consumption level $l(r)$ for each resource r at the end of the corresponding partial path. L_0 is set to a unique label l_0 representing the partial path reduced to i_0 with resource consumptions $l_0(r) = 0$, $\forall r = 0, \ldots, R$. At iteration q, label set L_{q+1} is computed from L_q by extending each label $l \in L(q)$ and by applying dominance rules to reduce the cardinality of L_{q+1}. Extension of a label $l \in L_q$ consists in creating a label l' for each outgoing arcs $(i_q, i_{q+1})^e$ of i_q. Resource consumptions are updated as follows: $l'(r) = l(r) + d^e_{q,q+1}(r)$, $\forall r = 0, \ldots, R$; l' is discarded once there is a resource r such that $l'(r) > Q^r$ or if it is dominated by another label of L_{q+1}. Since all labels of L_q visit the same nodes from i_0 to i_q, a label $l \in L_q$ dominates another label $l' \in L_q$ if $l(r) \leq l'(r), \forall r = 0, \ldots, R$. At iteration $N-1$, the dominant set L_N, which includes the optimal solution, is computed.

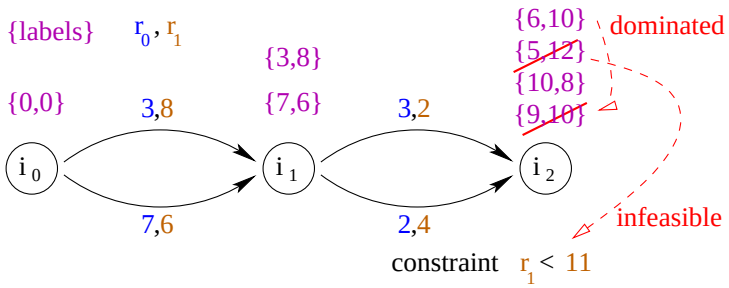


Fig. 3. Dynamic programming algorithm for FSASP

Note that unlike Bellman's algorithm, that can be applied when no resource is 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. More precisely, we can compute an upper bound on the number of non-dominated labels in each set L_q. Indeed if resource consumptions and availabilities are integer we have $|L_q| \leq \Lambda$ where $\Lambda = \Pi_{r=1,\ldots R} Q^r$ is the number of distinct consumption vectors. At each iteration q and for each label $l \in L_q$, all the outgoing arcs of i_q are enumerated. It follows that the three

Algorithm 1. Dynamic programming algorithm
Data: $G_{FS} = (V_{FS}, A_{FS})$ a linear multigraph corresponding to sequence $i_0, \ldots, i_N$
Result: L_N
initialization : $L_0 := 0_R$;
for: $q = 0$ to $N - 1$ **do**
  foreach: label $l \in L_q$ **do**
    foreach: outgoing arc a from i_q **do**
      $l' :=$ l extension from i_q to i_{q+1} by a;
      $dominated := false$;
      foreach: label l'' in L_{q+1} **do**
        if $l'(r) \leq l''(r), \forall r = 0, \ldots, R$ **then**
          $L_{q+1} := L_{q+1} \setminus \{l''\}$;
        else if $l''(r) \leq l'(r), \forall r = 0, \ldots, R$ **then**
          $dominated := true$;
          break;
        end
      end
      if *dominated*=false **then**
        $L_{q+1} := L_{q+1} \cup \{l'\}$;
      end
    end
  end
end

upper level loops are performed in $O(|A_{FS}|\Lambda)$. A label is extended in $O(R)$. Checking whether a label is dominated takes in turn $O(\Lambda(R))$ time. Consequently the algorithm has a worst-case complexity of $O(|A_{FS}|\Lambda^2 R)$. However, 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 relative small number of vertices), which helps finding optimal solutions efficiently (Irnich and Desaulniers, 2005).

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 infeasible. The algorithm would thus only consider the two remainder labels to continue the sequence.

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 to build multigraphs in our computational experiments, either from standard state-of-the-art benchmark instances or from geographic data (see Section 4.5).

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 neighbourhood and find the best neighbour solution with one execution of the dynamic programming algorithm. This possibility is illustrated for the request insertion operator in Section 4.3.2.

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. As we show in Section 4.4, for column generation approaches, the multigraph representation has an impact on the subproblem definition but a simple adaptation allows to solve it efficiently.

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. Indeed, it can be simply seen that in a fractional solution at least one vertex has two different successors. For instance, in column generation, binary decision variables represent the selection of complete routes. And 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.

4 Application to the dial-a-ride problem

4.1 Practical motivation and related work

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 reservation software for 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 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 time that cannot be exceeded. An acceptable quality of service can be ensured by providing a guarantee on the maximum gap between the pick-up time and that latest drop-off time. 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. Defining the vehicle routing problem that way permitted to satisfy the different partners involved in the system.

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 impacts 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 real-life 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, robust and particularly adapted to dynamic situations. The insertion procedure of Jaw *et al.* (1986) is a reference in the context of the DARP. It deals 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 time. 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 resultant 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 (2006) 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.* (2006) solve with local search a complex DARP including various customer's and driver's 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 contain a *scheduling* subproblem to determine service times.

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.*, 2007) 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 dynamic 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 (2006) solve small instances (less than 50 requests). Ropke *et al.* (2007) 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 and 2007). Desaulniers *et al.* (2002) and Crainic and Laporte (1998) present more general information on *Pickup & Delivery Problems* and other vehicle routing problems. In view of the importance of insertion and branch-and-price methods for the solution of the DARP, we consider these approaches for the examination of the multigraph representation in this paper.

4.2 Problem Formulation

The problem is to serve a set $\mathcal{R}$ of requests with a heterogeneous fleet K of vehicles, where K is composed of subsets K^v of vehicles of identical type[1] $v \in \mathcal{VT}$. 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 time B_{r^-} and the maximum gap δ_r to respect between B_{r^-} and the actual pick-up time. Each vehicle k is characterized by its capacity C_k, starting and 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. Moreover, a pick-up or a drop-off point is called a service. Besides the characteristics defined above, a service i has a non-negative duration s_i.

Let $G = (V, A)$ be a directed graph. 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 $(i, j)^e$. Arcs with index $e = 0$ represent min-time road-paths.

The latest drop-off and maximum gap constraints can be expressed as time

[1] Vehicles are said of identical type if they share the same characteristics, including depots.

windows, as can be seen in equations (5)-(7):

$$B_{r^+} = B_{r^-} - d^0_{r^+r^-}(2) - s_{r^+} \tag{5}$$
$$A_{r^+} = B_{r^-} - \delta_r - s_{r^+} \tag{6}$$
$$A_{r^-} = A_{r^+} + s_{r^+} + d^0_{r^+r^-}(2) \tag{7}$$

where A_i is the earliest starting time for service i and B_i the latest starting time. 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 a *route*. It is called a *sequence*, if the service times 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 times 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:

$$\min \sum_{k\in K} \sum_{(i,j)^e\in A} x^k_{ije} d^e_{ij}(0) + \sum_{k\in K} \sum_{(o_k,i)^e\in A\setminus\{(o_k,m_k)^e\}} pc_k x^k_{o_kie} \tag{8}$$

subject to

$$\sum_{k\in K} \sum_{(r^+,j)^e\in A} x^k_{r^+je} = 1 \qquad \forall r \in \mathcal{R}, \tag{9}$$
$$\sum_{(o_k,j)^e\in A} x^k_{o_kje} = 1 \qquad \forall k \in K, \tag{10}$$
$$\sum_{(r^+,j)^e\in A} x^k_{r^+je} - \sum_{(j,r^-)^e\in A} x^k_{jr^-e} = 0 \qquad \forall k \in K, \forall r \in \mathcal{R}, \tag{11}$$
$$\sum_{(i,j)^e\in A} x^k_{ije} - \sum_{(j,i)^e\in A} x^k_{jie} = 0 \qquad \forall k \in K, \forall j \in V, \tag{12}$$
$$x^k_{ije}\left(T_i + s_i + d^e_{ij}(2) - T_j\right) \le 0 \qquad \forall k \in K, \forall\,(i,j)^e \in A, \tag{13}$$
$$A_i \le T_i \le B_i \qquad \forall k \in K, \forall i \in V, \tag{14}$$
$$T_{r^+} + s_{r^+} + d^0_{r^+r^-}(2) \le T_{r^-} \qquad \forall k \in K, \forall r \in \mathcal{R}, \tag{15}$$
$$x^k_{ije}\left(L_i + d^e_{ij}(1) - L_j\right) = 0 \qquad \forall k \in K, \forall\,(i,j)^e \in A, \tag{16}$$
$$L_{r^+} \sum_{(r^+,j)^e\in A} x^k_{r^+je} \le C_k \qquad \forall k \in K, \forall r \in \mathcal{R}, \tag{17}$$

$$x^k_{i^+j^-e} = 0 \, \forall k \in K, \left(i^+, j^-\right)^e \in A : d^e_{i^+j^-}(2) = 0 \quad (18)$$

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

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

Objective function (8) involves minimization of the cumulative cost of the selected arcs plus the total fixed cost of the employed vehicles.

Constraints (9) enforce that exactly one vehicle passes through exactly one arc $(r^+, j)^e$ for each request r. This coresponds to the fact that each pick-up request must be fulfilled by exactly one vehicle. Constraints (10) ensure that vehicles leave their starting depot exactly once, possibly for a dummy run to their arrival 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. Constraints (13) enforce the precedence constraint between the service time 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) express the load conservation at each service, respectively. Constraints (17) ensure that a vehicle leaving a pick-up service has the required capacity. On the same station, pick-ups occur after drop-offs (constraints (18)). Note that non-linear constraints (13) and (16) and (17) could be presented as linear constraints at the expense of "big M" coefficients. Since we do not use directly the model to solve the problem or even its linear relaxation, the non-linear model was chosen for the sake of clarity.

The objective function was defined in accordance with the Doubs Central Transportation Organizing Authorities so as to minimize, as a main objective, the cost incurred while making the service more attractive via a secondary hierarchical objective. This secondary objective is the minimization of the time lost by users. We express that as the sum of the gaps between the latest drop-off times and the actual pick-up times (see equation (20)).

$$\min \sum_{r \in \mathcal{R}} (B_{r^-} - T_{r^+}) \quad (20)$$

This optimization does not change the fixed sequences (x^k_{ije} variables) computed first. In addition, we consider the following constraint on service times: a vehicle is allowed to wait at a stop only after a drop-off followed by a pick-up. Adding this constraint to the model (8-19) preserves its optimal value.

4.3 Insertion heuristic

We propose a five-step insertion-based heuristic illustrated in Figure 4.

(1) The first step is a greedy insertion procedure which aims at constructing vehicle sequences satisfying all requests.
(2) A descent method, based on removals and insertions, is used in the second step to improve the set of sequences.
(3) In the third step, the greedy insertion procedure is called again with a different request order based on marginal costs.
(4) The descent method of Step 2 is used once more at the fourth step on the solution obtained at Step 3.
(5) The sequences are scheduled (time-stamped), and the routes obtained, during Step 5.

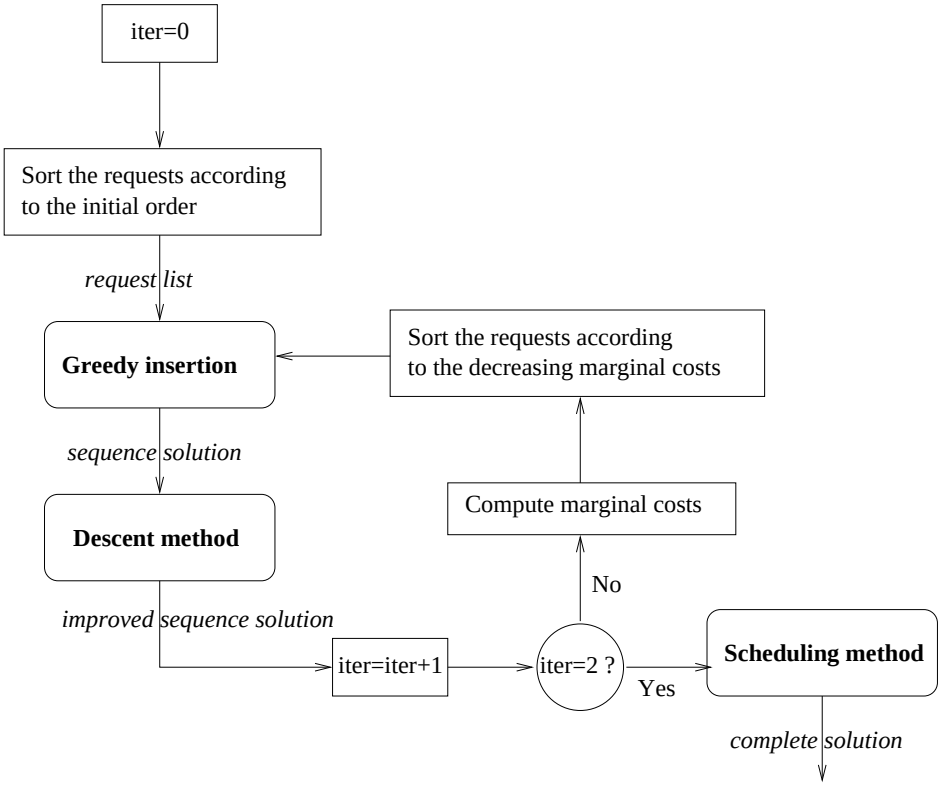


Fig. 4. The five-step insertion-based heuristic

The main objective is tackled during steps 1 to 4 while the secondary objective is ignored. Step 5 optimizes the secondary objective while keeping the best value found for the main one. Section 4.3.1 presents more precisely the different steps of the algorithm. Section 4.3.2 details the request insertion mechanism as a special FSASP problem.

4.3.1 The five-step approach

4.3.1.1 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. Least-cost insertion thus means in this context that the insertion chosen is the one that provides a feasible sequence and a provisional related arc selection with a minimal cost. 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 Section 4.3.2.

For the first call to this phase, an initial order of requests is defined on increasing values of the earliest pick-up time. The order used for the second call to this phase (Step 3 of the 5-step heuristic) relies on a marginal cost mechanism. This cost is calculated by removing the request temporarily and by evaluating the corresponding profit, which requires solving an FSASP on the new sequence. The order is then defined as the decreasing order of marginal costs, so that the most expensive requests are inserted first.

4.3.1.2 Descent method The descent method considers the set of computed sequences and tries to move requests. Requests are removed and reinserted with a least-cost policy. A first time, we select all the requests according to the order used to build the solution. Then, we select each request to remove and reinsert randomly until no more improvement is possible. Insertions still imply to solve a FSASP as explained in Section 4.3.2. Evaluating the cost of a sequence after the removal of a request also implies arc re-oprimization.

4.3.1.3 Scheduling (time-stamping) At the end of Step 4, a set of sequences is available. The optimal selection of arcs for these sequences is given, with the guarantee that a feasible schedule exists. The first hierarchical level of the objective function is then fixed. We denote S one of these sequences. We optimize service times according to the second hierarchical level for S, *i.e.*, minimize the sum of the gaps between latest drop-off times (B_{r^-}) and pick-up times (T_{r^+}). This amounts to maximize the following objective (21) since the latest drop-off times are constant:

$$\max \sum_{r \in S} T_{r^+} \tag{21}$$

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 time 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 (22-23), calculates the so-called *Latest Pick-Up Earliest Drop-Off Solution.* The first iteration (22) computes the latest feasible service times that characterize the *Latest Scheduling Solution.* $(i,j)^e$ denotes without any ambiguity the selected arc between i and j. For the sake of simplicity, we consider here that sequence S is sequence $\{1,\ldots,|S|\}$. By construction, we state easily that this schedule is optimal for the secondary criterion.

$$T_{|S|} := B_{|S|};\ T_i := \min\left\{B_i, B_{i+1} - d^e_{i,i+1}(2) - s_i\right\} \forall i = |S|-1,\ldots,1; \quad (22)$$

$$T_i := \max\left\{A_i, T_{i-1} + s_{i-1} + d^e_{i-1,i}(2)\right\} \quad \forall i \text{ a drop-off.} \quad (23)$$

The *Latest Pick-Up Earliest Drop-Off Solutions* respecting these operational constraints form a dominant set for the routing problem and the scheduling problem.

4.3.2 Request insertion in a sequence

The insertion of a request r in a sequence (Step 1 and Step 3 of the 5-step heuristic) basically relies on the solution of a FSASP (see Section 3.1) in an augmented graph. 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 every arc respecting the sequence order (Figure 5). Initial constraints or constraints induced by the sequence structure can immediately reduce the number of insertion positions.

Finding the best insertion position for r is equivalent to finding 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

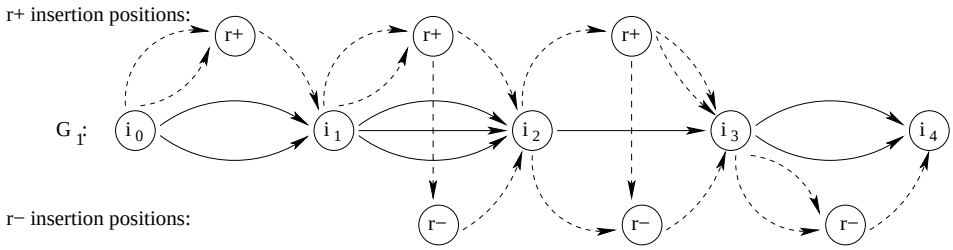


Fig. 5. Request insertion in a sequence – multigraph construction

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 1 (with the resulting label being lb'). The rules for resource R are summarized in Table 2 depending on the destination vertex. The R^{lb} starting level at the depot is set to 0.

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

Table 1
Extension functions for resources T, L and C

arc	(i,j)	(i,r^+)	(i,r^-)	$(i,depot)$
value of $R^{lb'}$	R^{lb}	$R^{lb}+1$	$R^{lb}+1$	R^{lb}
constraint	$-$	$=1$	$=2$	$=2$

Table 2
Extension function for resource R

These rules forbid that drop-off occurs before pick-up and imply that each of the two services is inserted exactly once. 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.

$$C^{lb_1} \leq C^{lb_2}; T^{lb_1} \leq T^{lb_2}; (R^{lb_1} = R^{lb_2}\, or\, R^{lb_1} = 2) \tag{24}$$

This condition also controls the loads L^{lb_1} and L^{lb_2} as we show in Table 3. The load of lb_1 never exceeds the load of lb_2 if the constraint on R is satisfied.

In addition, when the sum of the new request load and the maximal load in the sequence does not exceed the vehicle capacity, the case $R^{lb_1} = 1$ and $R^{lb_2} = 0$

$R^{lb_1} \backslash R^{lb_2}$	0	1	2
0	$L^{lb_1} = L^{lb_2}$	$L^{lb_1} \leq L^{lb_2}$	$L^{lb_1} = L^{lb_2}$
1	$L^{lb_1} \geq L^{lb_2}$	$L^{lb_1} = L^{lb_2}$	$L^{lb_1} \geq L^{lb_2}$
2	$L^{lb_1} = L^{lb_2}$	$L^{lb_1} \leq L^{lb_2}$	$L^{lb_1} = L^{lb_2}$

Table 3
Constraints on loads of two comparable labels

can be ignored yielding a stronger dominance rule (25).

$$C^{lb_1} \leq C^{lb_2}; T^{lb_1} \leq T^{lb_2}; R^{lb_1} \geq R^{lb_2} \tag{25}$$

We discussed the complexity of the Algorithm 1 in Section 3.1. If we assume that service times have integer values, we obtain for an insertion in G_1 a worst-case complexity of $\mathcal{O}(|A_1| \times (\max_{1 \leq i \leq N} \{B_i\})^2)$. We deduce a worst-case complexity of $\mathcal{O}(|\mathcal{R}||A| \times (\max_{1 \leq i \leq N} \{B_i\})^2)$ for the complete insertion procedure.

Once the best insertion is found, a basic constraint propagation procedure on time windows is applied to improve the efficiency of further insertions. The *Latest Scheduling Solution* (described in Section 4.3.1.3, equation (22)) computed with min-time arcs ($e = 0$) gives new bounds B_i and the *Earliest Scheduling Solution* – obtained by a symmetric construction – gives new bounds A_i. This update is made traversing the sequence twice (one time in each direction) in $\mathcal{O}(|V_1|)$. All feasible solutions are preserved.

4.4 Exact method

In this section, we present a branch-and-price exact solution procedure for the considered DARP. This scheme was first introduced for the DARP in Dumas *et al.* (1989). The adaptation to the multigraph case is described subsequently. The reader is referred to Desaulniers *et al.* (2005) for more details on column generation techniques.

Let Ω_v be the set of possible time-stamped routes for a vehicle of type $v \in \mathcal{VT}$ carrying out at most once each potential transportation request, satisfying the time windows and capacity constraints. Let $\Omega = \bigcup_{v \in \mathcal{VT}} \Omega_v = \{\omega_1, \ldots, \omega_{|\Omega|}\}$ be the complete set of possible routes (identical routes assigned to different vehicle types are considered different). Let $b_{vn} = 1$ if route $\omega_n \in \Omega_v$, $b_{vn} = 0$ otherwise. Let $a_{rn} = 1$ if route $\omega_n \in \Omega$ carries out request r, $a_{rn} = 0$ otherwise. Let c_n^1 be the total costs generated by route $\omega_n \in \Omega$. The DARP can be stated

as follows:

$$\min \sum_{\omega_n \in \Omega} c_n^1 \lambda_n \tag{26}$$

subject to

$$\sum_{\omega_n \in \Omega} a_{rn} \lambda_n = 1 \ \ \forall r \in \mathcal{R}, \tag{27}$$

$$\sum_{\omega_n \in \Omega} b_{vn} \lambda_n \leq |K^v| \, \forall v \in \mathcal{VT}, \tag{28}$$

$$\lambda_n \text{ integer } \forall \omega_n \in \Omega. \tag{29}$$

The decision variables λ_n indicate whether route $\omega_n \in \Omega$ is used or not. Objective function (26) corresponds to objective function (8) in the compact model. Constraints (27) ensure that each request is carried out exactly once. Constraint (28) limits the number of vehicles of type v used to $|K^v|$. We denote by π_r the dual variable associated with constraint (27) for request r and by μ_v the dual variable associated with constraint (28) for vehicle type v.

Solving the linear relaxation of model (26)-(29) necessitates the use of a column generation technique, due to the size of Ω. In the following, we call Master Problem (MP) the linear relaxation of model (26)-(29).

Column generation is based on two components: a restricted master problem and one or more subproblems. The restricted master problem MP(Ω_1) is obtained from MP by considering only a subset $\Omega_1 \subset \Omega$ of variables. A subproblem aims at adding progressively new potentially good columns to Ω_1 until an optimality criterion is attained. One consider distinct subproblems for each vehicle type.

Ω_1 is initialized with a simple set of routes, for instances routes obtained by the insertion heuristic of the Section 4.3. At each iteration of the algorithm, MP(Ω_1) is solved with the simplex method. A subproblem determines for a specific vehicle type v whether some variables λ_n with $\omega_n \in \Omega_v$ have a negative reduced cost. This condition can easily be stated as:

$$c_n^1 - \sum_{r \in \mathcal{R}} a_{rn} \pi_r - \mu_v \leq 0. \tag{30}$$

One or several variables with negative reduced cost are then added to Ω_1 and the algorithm iterates until all subproblems fail to find new routes.

A subproblem, for a vehicle type v can be seen as an *Elementary Shortest Path Problem with Resource Constraints* (ESPPRC) on graph G, aiming at

finding an elementary path of minimal cost subject to resource constraints between the starting and arrival depots of vehicle type v. The path has to be elementary in the sense that requests should not be carried out more that once. Resources and cost are defined with the following $|\mathcal{R}| + 3$ attributes: resources L and T for the load and time consumption, cost C, one resource for every request telling about the presence of the request in the path.

Extension functions and violation rules on attributes L, T and C are defined exactly as explained in Table 1, except that cost matrix $d^e_{ij}(0)$ is changed to matrix $dr^e_{ij}(0)$ defined as explained below:

$$dr^e_{ij}(0) = d^e_{ij}(0) - \begin{cases} \pi_r \text{ if } j \text{ is the pick-up service of a request } r, \\ \mu_v \text{ if } j \text{ is the depot for vehicles of type } v, \\ 0 \quad \text{otherwise.} \end{cases} \tag{31}$$

With this new definition of arc costs, the cost of a path corresponds to the reduced cost value for this path.

Extension functions and violation rules on the $|\mathcal{R}|$ other attributes also behave as explained in Table 2, except that value 0 is authorized when extending the label to the depot (see Table 4). Indeed, a request does not necessarily belong to the solution, contrary to the situation of Section 4.3.2. In Table 4, R^{lb} represents the value of the resource for a label lb and $R^{lb'}$ the value of this resource for the label lb' obtained after extension, depending on the type of arc used for the extension.

arc	(i,j)	(i,r^+)	(i,r^-)	$(i,depot)$
value of $R^{lb'}$	R^{lb}	$R^{lb}+1$	$R^{lb}+1$	R^{lb}
constraint	$-$	$=1$	$=2$	$\in\{0,2\}$

Table 4
Extension function for the resource R associated to request r

The dominance rule works as follow. lb_1 dominates lb_2 if equations (32) are valid, where labels lb_1 and lb_2 represent two paths from the depot to the same current ending vertex i. Label lb_2 is penalized if it is more expensive and longer (in duration) than lb_1 and if it has to close an opened request which is not opened for lb_1. Rules on request resources include rule on loads.

$$\begin{aligned} &C^{lb_1} \leq C^{lb_2}; T^{lb_1} \leq T^{lb_2}; L^{lb_1} \leq L^{lb_2}; \\ &R^{lb_1}_r = 0 \text{ or } R^{lb_2}_r = 1 \text{ or } (R^{lb_1}_r = 2 \text{ and } r \in Unreachable(lb_2)) \; \forall r = 1, \ldots, |\mathcal{R}| \end{aligned} \tag{32}$$

The formula (33) defines the set of unreachable requests for a label lb ending in vertex i.

$$Unreachable(lb) = \{r \in \mathcal{R} : R_r^{lb} > 0\} \cup \{r \in \mathcal{R} : T^{lb} + T_{ir^+} + T_{r^+r^-} > B_{r^-}\} \quad (33)$$

The ESPPRC is solved through a dynamic programming approach, as proposed in Feillet *et al.* (2004). Note that the presence of multiple arcs between pairs of vertices only implies to extend every label along each one of these arcs (resulting in one label per arc). Note also that this algorithm can be seen as an extension of Algorithm 1 presented in Section 3.1, allowing the solution of the problem in a acyclic graph. The single modification needed compared to Algorithm 1 is that the main loop is repeated until every label has been extended (keeping that labels are extended only once). The subproblems are not necesseraly solved to optimality. The purpose is to find out routes with negative reduced cost and therefore we stop when we compute enough such routes.

The branching scheme consists, as described in Section 3.2, in enforcing or forbidding the complete set of arcs between two vertices. These constraints are easy to handle at the master problem level by removing incorrect columns and are classically transferred to the subproblem by removing appropriate arcs. Note that the scheduling procedure (the fifth step of the heuristic) can still be applied to improve the solution in the ODT context.

4.5 Results

Due to 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.

We generated 24 benchmark instances from geographical data of Doubs Central (using IGN [2] maps) and estimated flows of population. Flows were generated according to 4 scenarii: a random flow, a convergent flow, an extrem convergent flow and a multiconvergent flow denoted *'rand'*, *'conv'*, *'econv'* and *'mconv'*, respectively. For each scenario, 6 instances with 25, 50 and 100 requests were generated. The maximum gap between the latest drop-off time and the actual pick-up time ($B_{r^-} - T_{r^+}$) is 1.5 or 1.3 times longer than the min-time path (in the road network). The fleet is heterogeneous and corresponds to taxi companies with 8 depots. All vehicles are 6-seater similar cars are charging a fixed cost $pc_k = 10$.

[2] Institut Géographique National

We also have new instances derived from the literature. Cordeau generated two series (*'a'* and *'b'*) of random Euclidean (on a [20x20] square) DARP instances, described in Cordeau (2006). The instances are named as xy_z where x is either 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 time. Since all requests have the same maximal individual ride time and the same time windows size. There is a drawback 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 (only from the *'b'* series) a more time constrained one denoted by series *'c'*, 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 bigger instances (*'C'* ones) by concatenation of some Cordeau instances. From 16 to 630 requests must be served by an unlimited homogeneous fleet. These multigraph series were produced by adding arcs and computing Pareto optimal paths through dynamic programming. We added $|V|^2 \times 10\%$ arcs ($|V|$ is the number of vertices) for all series except for the 'C-630' instance with which only 2% arcs were added. Additional arcs are slower (according to attribute $d^e_{ij}(2)$) than initial arcs between 20% and 40%; they are cheaper (attribute $d^e_{ij}(0)$) in the same proportions. Thus we construct road networks such as the time and cost deviations of an arc from the min-time arc, does not exceed 40%. This maximal gap is equal to 50% in the Doubs Central road network. Euclideans multigraphs cover a larger interval of values of densities ($[0.39, 1.16]$) than the Doubs Central's ones ($[0.45, 0.58]$). These instances and the characteristics of the multigraphs are more deeply detailed in Annex A.

The results obtained with the algorithms presented in Sections 4.3 and 4.4 on multigraph and those obtained on min-time paths simple graph, are compared in the Tables 5 and 6 for the Euclidean instances and in Table 7 and 8 for the realistics ones. Italic typography style is used to highlight the results obtained on multigraphs. Each instance is solved with the 5-steps heuristic. Then, computed routes initiate the column generation algorithm. For each series we indicate the total cost (*'cost'* in Tables 5 and 7) of the current solution at different steps of the insertion heuristic (steps 1, 2 and 5) and after the exact solution. Values for the optimals solutions of the second objective, the time lost, and the number of vehicles can be found in columns *'time lost'* and *'vehicles'*. At the first and the final steps, we compute the gap between the current solution and the optimal solution obtained on the simple graph. The row *'gap'* indicates the average of this gap on all instances above: $gap = (\text{Cost} -$

Optimal Cost on Simple Graph) $\times$ 100/Optimal Cost on Simple Graph. Computing times (*'cpu time sec.'* in Tables 5 and 7) are given in seconds; the computing time of the optimal solution cumulates the heuristic and the column generation computing times. The quantities of FSASP solved by the heuristic are given by columns *'insertions'* in Tables 6 and 8, in the request insertion case, and *'arc selections'* in Tables 6 and 8 in the request deletion case. Note that in the case of the simple graph, the FSASP solved only aims at computing the schedule of services and checking feasibility. Columns *'ESPPRC'* and *'nodes'* represent the number of subproblems solved and the number of nodes created during the branch-and-price algorithm, respectively.

The exact method is only able to solve instances with less than 100 requests which is a reasonable performance; the low number of explored nodes shows the high quality of the linear relaxation of the master problem. However, its purpose was actually to evaluate the quality of the proposed heuristic. It comes out that the obtained heuristic solutions are between 5% and 10% worse than optimal ones in average, either on a simple graph or on a multigraph and either on Euclidean or on realistic instances. Additional experiments show that running the heuristic initialized with different initial orders (even randomly generated) reduces this gap. Our insertion heuristic is very time consuming for solving the biggest instances, the 'C' series; the first step runs about one hour for the 'C-310' instance! Passenger flows are less than 100 requests for a day in the Doubs Central. Consequentely, the heuristic performance was considered as satisfactory for the Doubs Central ODT application.

Using a multigraph reduces costs by more than 15% for Euclidean instances and 10% for realistic instances, which is considerable for poorly dense road network of a rural zone. By using a multigraph , better (about 1% and 14% in average for realistic and Euclidean instances, respectively) results with the heuristic than the exact approach using the simple graph even can be obtained. On many instances, the initial greedy procedure (*'step 1'*) is enough to out perform the exact method on the simple graph. As expected, the time lost by passengers could increase dramatically by using a multigraph. Alternative paths allow the transporter to go further in the exploitation of customers' time constraints. However, this impact is correlated to the flow structure. For instance, the time lost is not penalized for *'econv'* series. The most surprising result is the relatively low number of vehicles used by solutions on multigraphs. It means that a non negligible interesting (and time feasible) routes mergings are rejected from the solutions on simple graphs because of their high costs. The similar results obtained on *'1.3 delayed'* and on *'1.5 delayed'* realistic instances show the positive impact of the multigraph model even on very time constrained instances. The heterogeneous fleet of realistic instances causes a large number of FSASP (*'insertions'*) in the heuristic case and slave problem (*'ESPPRC'*) in the exact method case and thus long computing times.

In order to link together the cost reduction on solutions and some characterisitcs of the multigraph, we compute the graph of the Figure (6). For each instance, the *'mincost'* curve, which represents the average cost savings between min-cost and min-time arcs, is drawn as a function of the *'gap'* between simple and multi- graph solutions. The *'gap'* tends to increase with high values of the *'mincost'*. The dotted line represents the density of the multigraph obtained after a set of trivial reductions according to time constraints. The density of multigraphs also increases with the *'gap'*. We see in the column *'mincost'* of the Tables 6 and 8 that the optimal solutions on multigraphs use a high rate of min-cost arcs and min-time arcs. According to these values, we guess that *'mincost'* line is strongly correlated to the *'gap'*. By construction, our instances with high values of *'mincost'* tend to have high density. Almost all irregularities of these curves come from some realistic instances with a strong polarity which mitigates the effect stated above.

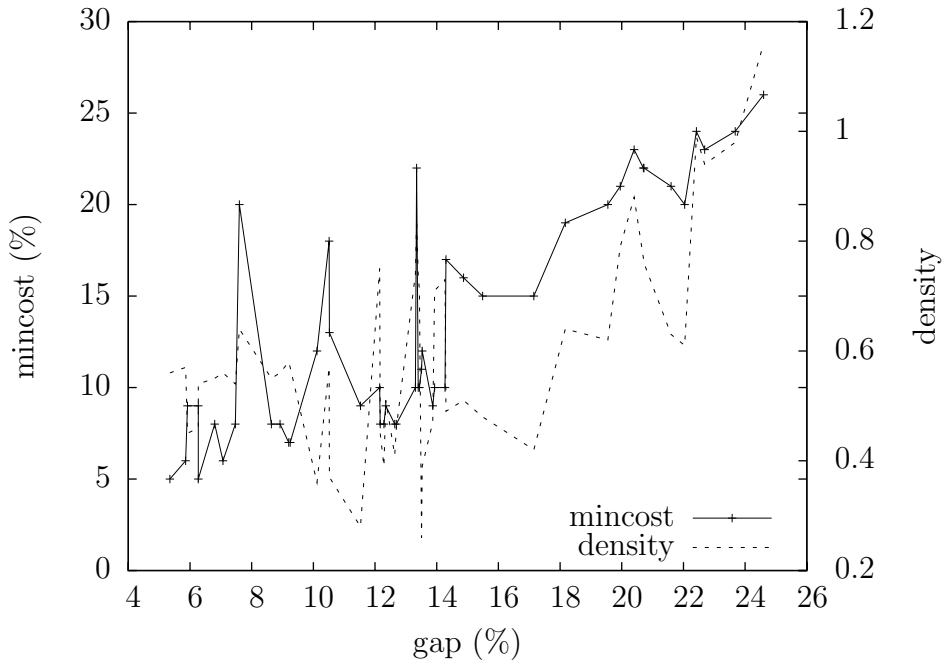


Fig. 6. Multigraphs gains and attributes

These results show a strong impact of multigraph on both cost and CPU consumption. A reservation software, jointly developed by the project team (UMR ESPACE, LIA) named TADOU has been issued from this research incorporating the proposed heuristic and is currently operational in the Doubs Central area. These tests were carried out with a 2.3 GHz Intel and 512 Mo RAM and a 3600 seconds CPU time limit. The application is a C++ compiled kernel of a Microsoft Visual Basic application embedded in a database (Microsoft Access) environment.

instance	cost						time lost		vehicles		cpu time sec.					
name	*step 1*		*steps 1-5*		*exact*		*exact*		*exact*		*step 1*		*steps 1-5*		*exact*	
c2-16	343.6	*293.8*	332.1	*292.0*	320.2	*276.8*	4	*13*	3	*3*	0	*0*	0	*0*	0	*0*
c2-20	355.8	*314.2*	345.3	*310.9*	339.8	*305.4*	6	*9*	4	*4*	0	*0*	0	*0*	0	*0*
c2-24	478.6	*391.1*	453.1	*372.5*	440.8	*365.2*	3	*22*	3	*3*	0	*0*	0	*0*	0	*1*
c3-18	350.2	*306.2*	337.7	*292.9*	334.3	*289.2*	5	*10*	4	*4*	0	*0*	0	*0*	0	*0*
c3-24	456.2	*379.8*	421.0	*359.7*	401.6	*341.9*	12	*29*	6	*6*	0	*0*	0	*0*	0	*0*
c3-30	587.9	*499.5*	562.1	*471.1*	543.0	*458.9*	7	*25*	5	*4*	0	*0*	0	*0*	0	*1*
c3-36	617.5	*518.5*	617.5	*506.4*	554.5	*496.3*	4	*26*	6	*5*	0	*0*	0	*1*	0	*4*
c4-16	335.7	*299.0*	332.6	*299.0*	327.3	*289.6*	4	*11*	4	*4*	0	*0*	0	*0*	0	*0*
c4-24	462.8	*396.4*	413.7	*369.6*	407.7	*364.8*	14	*28*	5	*4*	0	*0*	0	*0*	0	*0*
c4-32	627.7	*499.9*	574.3	*488.5*	542.2	*464.7*	11	*24*	6	*5*	0	*0*	0	*1*	0	*1*
c4-40	719.4	*582.0*	697.1	*570.9*	673.2	*550.9*	2	*39*	6	*6*	0	*0*	0	*2*	1	*5*
c4-48	778.1	*664.8*	764.8	*649.2*	657.8	*607.8*	0	*55*	10	*7*	0	*0*	0	*3*	1	*12*
c5-40	742.9	*597.1*	731.1	*588.5*	716.7	*576.6*	13	*53*	5	*5*	0	*0*	0	*1*	1	*7*
c5-50	949.8	*749.2*	895.2	*693.3*	860.7	*674.8*	25	*89*	7	*6*	0	*0*	0	*4*	2	*12*
c5-60	1037.5	*860.2*	1033.4	*829.4*	926.6	*802.9*	5	*64*	11	*7*	0	*1*	0	*5*	2	*71*
c6-48	883.2	*674.3*	829.0	*648.8*	815.9	*636.0*	12	*51*	8	*8*	0	*0*	0	*2*	1	*7*
c6-60	1063.6	*856.6*	1001.9	*818.0*	961.4	*769.5*	16	*56*	7	*6*	0	*2*	0	*13*	6	*56*
c6-72	1213.0	*908.2*	1136.9	*881.4*	1073.1	*829.6*	20	*61*	8	*8*	0	*2*	1	*21*	26	*202*
c7-56	1052.7	*816.9*	1002.5	*793.2*	951.1	*754.2*	19	*57*	8	*8*	0	*1*	0	*4*	3	*19*
c7-70	1183.6	*889.3*	1078.5	*855.0*	1030.9	*820.6*	30	*72*	10	*9*	0	*1*	0	*10*	11	*82*
c7-84	1481.4	*1099.7*	1431.9	*1072.5*	1350.1	*1030.3*	22	*108*	9	*8*	0	*5*	1	*35*	58	*451*
c8-64	1116.1	*836.4*	1050.2	*823.1*	993.8	*787.8*	20	*53*	9	*9*	0	*1*	1	*5*	5	*48*
c8-80	1335.6	*1003.8*	1256.3	*967.4*	1169.4	*907.2*	38	*79*	9	*9*	0	*5*	1	*46*	41	*207*
c8-96	1477.9	*1116.9*	1468.1	*1097.5*	1383.5	*1043.2*	41	*114*	10	*9*	0	*9*	1	*86*	114	*1287*
gap (%)	10.1	*-11.0*	5.1	*-13.8*	0.0	*-17.0*										
C-168	2765.4	*1968.8*	2625.0	*1840.7*	2408.5	-	81	-	17	-	0	*96*	4	*639*	2379	-
C-310	4733.4	*3272.2*	4582.0	*3145.7*	-	-	-	-	-	-	1	*1364*	14	*7213*	-	-
C-630	8961.6	*6246.8*	8719.1	-	-	-	-	-	-	-	6	*4355*	119	-	-	-

Table 5
Results on euclidean instances

instance	insertions				*arc selections*		ESPPRC		nodes		arcs used in MG	
name	*step 1*		*steps 1-5*		*steps 1-5*		*exact*		*exact*		*mincost*	*mintime*
c2-16	51	*51*	649	*549*	112	*95*	11	*14*	1	*1*	85	12
c2-20	90	*90*	649	*558*	120	*100*	14	*17*	1	*1*	60	34
c2-24	72	*72*	862	*530*	189	*120*	25	*42*	1	*1*	45	52
c3-18	71	*71*	492	*496*	90	*90*	11	*13*	1	*1*	61	39
c3-24	114	*114*	1111	*949*	140	*116*	14	*18*	1	*1*	55	40
c3-30	131	*131*	993	*1462*	180	*240*	24	*54*	1	*1*	56	41
c3-36	475	*165*	3854	*1615*	133	*252*	21	*127*	1	*1*	61	36
c4-16	70	*70*	448	*445*	80	*80*	7	*10*	1	*1*	64	36
c4-24	96	*97*	912	*906*	144	*144*	11	*17*	1	*1*	83	10
c4-32	137	*137*	1606	*1326*	256	*224*	19	*54*	1	*1*	50	47
c4-40	216	*243*	3163	*2414*	396	*317*	33	*106*	1	*1*	44	54
c4-48	837	*249*	7729	*3748*	162	*480*	29	*180*	1	*1*	58	42
c5-40	225	*225*	1837	*1714*	280	*240*	37	*134*	1	*1*	67	29
c5-50	287	*292*	4163	*3773*	500	*445*	50	*151*	1	*1*	62	34
c5-60	1079	*427*	12949	*3795*	326	*420*	49	*458*	1	*1*	73	22
c6-48	349	*355*	3214	*3359*	336	*336*	32	*118*	1	*1*	72	25
c6-60	339	*339*	3124	*3327*	420	*420*	85	*379*	1	*1*	70	27
c6-72	463	*463*	5710	*5555*	648	*572*	135	*428*	1	*1*	72	24
c7-56	383	*351*	3550	*3166*	392	*392*	59	*199*	1	*1*	66	29
c7-70	526	*515*	5110	*4832*	490	*490*	99	*450*	1	*1*	75	24
c7-84	663	*699*	6832	*7191*	672	*672*	195	*825*	1	*1*	70	16
c8-64	442	*485*	7164	*4352*	704	*448*	76	*342*	1	*1*	76	21
c8-80	557	*572*	8925	*5557*	880	*560*	166	*530*	1	*1*	70	28
c8-96	892	*894*	7798	*12501*	670	*1056*	257	*1506*	1	*2*	77	21
C-168	2066	*2182*	27806	*27104*	1680	*1680*	1033	-	1	-	-	-
C-310	5923	*6061*	65275	*46918*	2790	*1923*	-	-		-	-	-
C-630	20810	*20164*	349593	-	8820	-	-	-		-	-	-

Table 6
Results on euclidean instances

instance	cost						time lost		vehicles		cpu time sec.					
name	*step 1*		*steps 1-5*		*exact*		*exact*		*exact*		*step 1*		*steps 1-5*		*exact*	
rand1.3-25	907.9	*817.9*	853.9	*781.1*	774.6	*728.7*	9	*29*	7	*6*	0	*0*	0	*0*	0	*3*
rand1.3-50	1863.5	*1715.4*	1771.9	*1618.3*	1603.3	*1464.8*	12	*42*	11	*11*	0	*0*	0	*5*	2	*40*
rand1.3-100	3440.9	*3130.6*	3230.2	*3007.4*	2892.7	*2676.5*	27	*79*	18	*19*	0	*1*	1	*16*	108	*785*
conv1.3-25	1051.4	*879.2*	884.2	*802.1*	814.2	*704.8*	0	*22*	8	*7*	0	*0*	0	*2*	0	*5*
conv1.3-100	1912.9	*1646.4*	1755.2	*1513.7*	1575.7	*1356.2*	33	*76*	9	*9*	0	*1*	0	*8*	5	*58*
conv1.3-50	3568.2	*2880.0*	3148.7	*2618.4*	2740.2	*2349.2*	85	*140*	13	*13*	0	*4*	1	*31*	82	*626*
econv1.3-25	1168.4	*1005.5*	1167.9	*1005.5*	1142.7	*984.2*	24	*23*	9	*9*	0	*0*	0	*1*	0	*2*
econv1.3-50	2176.5	*1886.5*	2170.4	*1886.5*	2134.2	*1864.5*	74	*92*	18	*18*	0	*0*	0	*3*	1	*13*
econv1.3-100	4195.6	*3656.1*	4120.7	*3624.0*	4008.5	*3515.9*	206	*208*	31	*31*	0	*2*	1	*9*	3	*77*
mconv1.3-25	641.4	*604.0*	570.1	*549.0*	528.0	*494.9*	3	*5*	5	*5*	0	*0*	0	*1*	0	*2*
mconv1.3-50	1162.2	*1163.8*	1051.3	*1010.9*	978.2	*909.0*	11	*16*	10	*10*	0	*0*	0	*3*	3	*10*
mconv1.3-100	2448.2	*2113.7*	2205.7	*1982.8*	1946.4	*1766.5*	0	*32*	13	*13*	0	*1*	2	*11*	94	*244*
random1.5-25	930.9	*834.7*	838.9	*804.0*	768.3	*720.1*	14	*53*	6	*6*	0	*0*	0	*1*	0	*3*
random1.5-50	1920.3	*1759.0*	1773.2	*1640.2*	1600.2	*1457.4*	18	*71*	11	*11*	0	*1*	0	*5*	3	*50*
random1.5-100	3327.8	*3069.2*	3134.0	*2866.8*	2751.8	*2564.6*	116	*182*	17	*16*	0	*2*	1	*18*	75	*266*
conv1.5-25	1003.0	*805.9*	867.1	*763.0*	776.7	*673.4*	32	*40*	7	*7*	0	*0*	0	*1*	0	*8*
conv1.5-50	1627.9	*1392.9*	1520.6	*1309.5*	1369.1	*1185.6*	121	*159*	6	*6*	0	*1*	0	*8*	8	*92*
conv1.5-100	3227.2	*2730.0*	2800.6	*2485.1*	2410.3	*2117.5*	246	*320*	11	*11*	0	*12*	1	*87*	323	*1881*
econv1.5-25	1041.4	*883.8*	1028.5	*862.0*	947.1	*830.1*	72	*84*	9	*9*	0	*0*	0	*1*	0	*3*
econv1.5-50	1961.8	*1677.8*	1869.8	*1671.9*	1752.0	*1538.8*	157	*195*	14	*14*	0	*1*	0	*5*	1	*37*
econv1.5-100	3762.6	*3179.6*	3484.6	*3070.8*	3272.5	*2857.1*	427	*490*	26	*25*	0	*3*	1	*16*	31	*1237*
mconv1.5-25	574.8	*540.4*	510.2	*498.2*	475.5	*450.1*	27	*31*	3	*3*	0	*0*	0	*1*	1	*3*
mconv1.5-50	1086.5	*1030.5*	1034.5	*964.3*	939.2	*884.1*	62	*59*	7	*7*	0	*0*	0	*2*	5	*14*
mconv1.5-100	2301.6	*2044.3*	2134.7	*1977.2*	1878.1*	*1705.2*	98	*92*	12	*12*	0	*2*	1	*12*	3600	*380*
gap (%)	18.7	*4.8*	9.5	*-1.1*	0.0	*-10.1*										
(*) After 3600 seconds the lower and upper bounds computed are 1877.45 and 1878.05																

Table 7
Results on Doubs Central instances

instance	insertions				*arc selections*		ESPPRC		nodes	
name	*step 1*		*steps 1-5*		*steps 1-5*		*exact*		*exact*	
rand1.3-25	303	*327*	2890	*2480*	166	*148*	175	*201*	5	*5*
rand1.3-50	758	*757*	11124	*9684*	594	*489*	133	*356*	1	*7*
rand1.3-100	2122	*2156*	27114	*37270*	996	*1390*	2374	*2307*	111	*91*
conv1.3-25	325	*304*	2503	*3235*	150	*194*	111	*162*	1	*1*
conv1.3-100	718	*711*	5918	*6764*	300	*349*	227	*476*	1	*1*
conv1.3-50	1833	*1739*	20547	*17819*	900	*798*	519	*915*	1	*1*
econv1.3-25	388	*395*	3254	*2243*	131	*92*	44	*42*	1	*1*
econv1.3-50	1061	*1072*	9385	*6528*	277	*186*	69	*122*	1	*1*
econv1.3-100	3364	*3403*	32043	*19550*	696	*393*	78	*222*	1	*1*
mconv1.3-25	289	*289*	2692	*2387*	169	*142*	55	*133*	1	*5*
mconv1.3-50	748	*746*	12892	*8653*	699	*445*	180	*148*	1	*1*
mconv1.3-100	1905	*1913*	27269	*21363*	1099	*898*	378	*607*	1	*1*
random1.5-25	310	*308*	3098	*3332*	168	*194*	99	*116*	1	*1*
random1.5-50	727	*731*	6790	*6764*	349	*344*	210	*355*	3	*7*
random1.5-100	2009	*2045*	22976	*26184*	896	*998*	1124	*698*	35	*5*
conv1.5-25	307	*286*	3278	*2330*	200	*150*	85	*210*	1	*1*
conv1.5-50	614	*617*	7113	*5414*	450	*348*	265	*487*	1	*1*
conv1.5-100	1641	*1572*	16355	*16555*	795	*799*	2943	*1093*	175	*1*
econv1.5-25	382	*388*	2138	*3206*	87	*129*	55	*45*	1	*1*
econv1.5-50	1021	*1018*	7111	*11178*	245	*388*	92	*213*	1	*1*
econv1.5-100	2902	*2924*	36422	*20664*	898	*492*	565	*912*	43	*35*
mconv1.5-25	297	*288*	2637	*2177*	174	*144*	121	*123*	1	*1*
mconv1.5-50	736	*734*	5647	*5482*	298	*295*	155	*178*	1	*1*
mconv1.5-100	1757	*1769*	21831	*16841*	893	*800*	43054	*804*	7416	*6*

Table 8
Results on Doubs Central instances

5 Conclusion

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

In the first step, we made 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. Known that this scheduling subproblem is NP-hard, we address it with a dynamic programming algorithm, based on a SPPRC modeling.

We then discussed how classical solution schemes, either based on local search (heuristics and metaheuristics) or enumeration (exact algorithms), can handle the multigraph representation.

An algorithm was derived to solve a DARP for an ODT system developed in the Doubs Central area in France. The insertion operator was specified for a request made of a pick-up and a drop-off service and an insertion-based heuristic made of a greedy constructive algorithm and a descent method is presented. To evaluate the performances of this heuristic, we developed an exact branch-and-price method. 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 both concerning the tractability of the multigraph representation and the savings in term of solution value that can be attained through this representation.

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

The 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. The 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 solving an SPPRC. Dealing with big instances necessitates certainly to consider suboptimal neighbourhoods in this phase, for instance by reducing the size of the multigraph. Focus on extremal arcs seems to be promising because of the large presence of these arcs in optimal solutions.

Concerning exact methods, column generation appears as a very natural tool

to cope with the multigraph representation. As we showed, a multigraph representation mainly impacts on the column generation subproblem, which happens to be a ESPPRC which can directly integrate the multigraph dimension. However, the results show that the cpu time dramatically increases for the multigraph representation, due to the difficulty of the so-defined subproblem. Hence a relevant issue is to define acceleration techniques taking account of the multigraph structure.

References

[1] 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.

[2] 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.

[3] 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.

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

[5] 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.

[6] 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.

[7] 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.

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

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

[10] 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.

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

[12] 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.

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

[14] 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.

[15] G. Desaulniers, J. Desrosiers, and M.M. Solomon, editors. *Column Generation*. Springer, 2005.

[16] 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.

[17] 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.

[18] 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.

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

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

[21] D. Feillet, P. Dejax, M. Gendreau, and C. Gueguen. An exact algorithm for the elementary shortest path problem with resource constraints: application to some vehicle routing problems. *Networks*, 44(3):216–229, 2004.

[22] 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.

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

[24] 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.

[25] 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.

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

[27] 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.

[28] 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.

[29] R.M. Jørgensen, J. Larsen, and K.B. Bergvinsdottir. Solving the dial-a-ride problem using genetic algorithms.

[30] D. Josselin and C. Genre-Grandpierre. Des transports à la demande pour répondre aux nouvelles formes de mobilité. le concept de modulobus. In B. Montulet *et al.*, editor, *Mobilités et temporalités*, pages 151–164. University of Saint-Louis, Bruxelles, 2005.

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

[32] 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.

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

[34] 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.

[35] 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.

[36] B. Rekiek, A. Delchambre, and H.A. Saleh. Handicapped person transportation: An application of the grouping genetic algorithm. *Engineering Application of Artificial Intelligence*, 19:511–520, 2006.

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

[38] 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.

[39] 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.

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

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

[42] 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.

[43] 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.

[44] 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.

[45] 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.
[46] P. Toth and D. Vigo. Heuristic algorithms for the handicapped persons transportation problem. *Transportation Science*, 31:60–71, 1997.
[47] A. Warburton. Approximation of pareto optima in multiple-objective, shortest-path problems. *Operations Research*, 35:70–79, 1987.
[48] 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

The Doubs Central has 170 stations and eigth depots. Each depot has an homogeneous fleet (considered as unlimited) of 6-seaters taxis which incurre 10 units fixed costs. In the random series (*'rand'*), requests start from any to any station. In the convergent (*'conv'*) and the extrem convergent (*'econv'*) series, request start from any station to a small number of stations. The requests of the extrem convergent scenario have strongly bounded maximum arrival times. About 75% of the requets concern the four most important (in the biggest cities) stations in the Doubs Central in the multiconvergent series (*'mconv'*). Numbers of passengers depend on estimated flows in Doubs Central.

The Table A.1 summarizes the parameters of the generated realistic instances.

Scenario\ Dimension	spatial	time	delay	requests
random	$n \to n$	[8 : 00, 18 : 00]	$1/3, 1/5$	$25, 50, 100$
convergent	$n \to 1$	[8 : 00, 18 : 00]	$1/3, 1/5$	$25, 50, 100$
extrem convergent	$n \to 1$	[13 : 00, 14 : 40]	$1/3, 1/5$	$25, 50, 100$
multiconvergent	$4 \to 4$	[8 : 00, 18 : 00]	$1/3, 1/5$	$25, 50, 100$

Table A.1
The four demand scenarii descriptions

The instances derived from the Cordeau's ones have an unlimited homogeneous fleet, in a central depot, of 6-seaters vehicles. Using one vehicle generates a fixed cost (=10). From outbound and inbound requests we keep the latest drop-off time and the earliest pick-up time, respectively. Then, we compute new time windows constraints using this time and a maximal ride time equal to 1.5 times the min-time path between the pick-up and the drop-off points. Requests concern up to 6 passengers and one half to less than 3.

The initial graph G_0 represents the road network between stops and depots. The reduced multigraph MG is the graph of services and non-dominated paths between services stops. An arc in G_0 generates one in MG, 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: $(i^+ \to i^- \to j^+ \to j^-)$; $(i^+ \to j^+ \to j^- \to i^-)$; $(i^+ \to j^+ \to i^- \to j^-)$; $(j^+ \to j^- \to i^+ \to i^-)$; $(j^+ \to i^+ \to i^- \to j^-)$; $(j^+ \to i^+ \to j^- \to i^-)$. The simple graph G is generated in the same way as MG but considering only min-time paths in G_0.

The Tables A.2 and A.3 summarize the main characteristics of the generated instances. In the Table A.2, Euclidean instances are grouped above the big instance (*'C-'*) they are contained in. The column *'mincost'* indicates the average percentage of cost savings between pairs of min-cost and min-time arcs. The column *'avgcost'* indicates the average of the average percentage of cost saving among all non dominated arcs between two vertices. Columns *'maxtime'* and *'avgtime'* give the same values for time increases. The columns $d0$, md and d show respectively the density of the road network, the multigraph MG and the simple graphs G. The graph density is computed as $d(G = (V, A)) = |A'|/(|V'||V' - 1|)$, where $G' = (V', A')$ is the subgraph of pick-up and drop-off (without depot). The density of MG can exceed 1 with this definition.

name	*mincost*	*maxtime*	*avgcost*	*avgtime*	$d0$	md	d
c2-16	12	18	6	9	2.79	0.39	0.13
c2-20	12	16	6	8	2.59	0.36	0.13
c2-24	15	17	7	9	3.53	0.42	0.13
c3-18	11	15	6	8	2.40	0.26	0.12
c3-24	16	20	8	10	4.50	0.51	0.12
c3-30	15	19	8	10	3.91	0.48	0.12
c3-36	18	20	9	10	4.88	0.57	0.12
C-168	30	23	15	11	13.55	1.58	0.12
c4-16	9	14	5	7	2.58	0.28	0.12
c4-24	13	17	6	8	3.38	0.37	0.12
c4-32	17	19	8	10	4.00	0.49	0.12
c4-40	19	20	9	10	5.45	0.64	0.12
c4-48	20	21	10	10	5.57	0.64	0.12
c5-40	20	22	9	11	5.56	0.62	0.12
c5-50	21	22	10	11	5.69	0.63	0.12
c5-60	22	23	10	11	7.19	0.84	0.12
c6-48	20	21	10	10	5.47	0.61	0.12
c6-60	21	21	10	10	6.85	0.79	0.12
c6-72	23	23	11	11	7.89	0.94	0.12
C-310	34	22	17	10	20.28	2.28	0.12
c7-56	22	23	11	11	6.29	0.77	0.12
c7-70	23	22	11	11	7.46	0.88	0.12
c7-84	24	23	12	11	8.33	0.98	0.12
c8-64	22	22	11	11	6.71	0.76	0.12
c8-80	24	22	12	10	8.60	0.99	0.12
c8-96	26	23	13	11	9.79	1.16	0.12
C-630	32	22	16	10	18.62	2.15	0.12

Table A.2
Instances characteristics

name	*mincost*	*maxtime*	*avgcost*	*avgtime*	$d0$	md	d
rand1.3-25	9	9	4	4	4.56	0.45	0.12
rand1.3-50	8	9	4	4	4.90	0.55	0.12
rand1.3-100	8	9	4	4	4.98	0.54	0.11
conv1.3-25	10	11	5	5	6.49	0.71	0.12
conv1.3-100	10	10	5	4	5.95	0.71	0.11
conv1.3-50	10	9	5	4	5.67	0.73	0.11
econv1.3-25	9	7	5	3	6.48	0.47	0.08
econv1.3-50	8	8	4	4	5.77	0.41	0.07
econv1.3-100	8	8	5	3	5.30	0.39	0.07
mconv1.3-25	5	8	3	4	6.80	0.54	0.12
mconv1.3-50	6	8	3	4	6.20	0.56	0.12
mconv1.3-100	7	8	4	4	5.96	0.57	0.11
random1.5-25	9	9	4	4	4.56	0.46	0.12
random1.5-50	8	9	4	4	4.90	0.56	0.12
random1.5-100	8	9	4	4	4.98	0.55	0.12
conv1.5-25	10	11	5	5	6.49	0.74	0.12
conv1.5-50	10	10	5	4	5.95	0.74	0.12
conv1.5-100	10	9	5	4	5.67	0.75	0.11
econv1.5-25	9	7	5	3	6.48	0.51	0.09
econv1.5-50	8	8	4	3	5.77	0.45	0.09
econv1.5-100	8	7	4	3	5.30	0.46	0.09
mconv1.5-25	5	7	3	3	6.80	0.56	0.13
mconv1.5-50	6	8	3	4	6.20	0.57	0.12
mconv1.5-100	7	7	4	4	5.96	0.58	0.12

Table A.3
Instances characteristics