Vehicle routing problems with alternative paths: an application to demand responsive transports

Thierry Garaix, Christian Artigues, Dominique Feillet, Didier Josselin

Vehicle routing problems with alternative paths : an application to demand responsive transports

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

October 13, 2006

Abstract

The class of vehicle routing problems involves the optimization of freight or person transportation activities. These problems are generally treated via the representation of geographical data as a valued complete graph. Every arc of the graph represents the shortest path for a possible origin-destination connection. Several attributes can be defined for an arc (travel time, travel cost...), but the path implied 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 dismissed at once from the solution space. In this work, we propose to represent geographic data 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 demand responsive transportation system developed in a French department. Computational experiments on realistic data underline the potential cost savings brought by the multigraph model.

KEYWORDS: *vehicle routing problem, demand responsive transportation, multigraph, shortest path problem with resource constraints.*

1 Introduction

The class of vehicle routing problems draws many researchers and industrial practitioners attention during the last decades. These problems involve the optimization of freight or person transportation activities. They are generally treated via the representation of geographic data as a valued complete graph. The vertex set is limited to the set of origin or destination points. Any arc of the graph then represents the shortest

path between two vertices. Several attributes can be defined for an arc (travel time, travel cost...), but the path implied 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 dismissed at once from the solution space, as in Figure 1. This can be problematic in many situations. A typical exam-

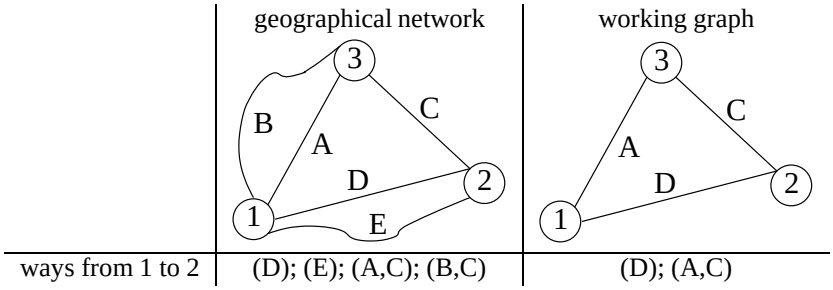

Figure 1: Simple graph construction

ple is provided by demand responsive transportation (DRT) systems. In such systems, a set of customer origin and destination locations is identified and vehicle routes are defined to transport the customers, according to some quality-of-service constraints and/or objectives. Though the path retained between two (origin or destination) customer locations is generally set as the shortest path in time, the driver or the shipper might prefer a less costly itinerary if he gets time. If the customer pays according to the distance (which is generally not the case in DRT systems, but is true in taxis), avoiding fast but long-distance Sections could also be of interest (for the customer). Note that initially 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 a DRT system implemented in the Doubs Central area in France.

In this work, we propose to represent geographic data with a multigraph, so that alternative routes are considered. Ideally, an arc will be added between two vertices for each Pareto optimal path according to arc attributes in the geographic network, unless they traverse another origin or destination location. Any good path of the geographic network 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. The multigraph representation then 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). A second situation would be met by a traveler having a tourist interest in his travel. One might then have some clearly identified destination points and different possibilities (with different duration and tourist interests) of linking these points. Actually, having a multigraph representation makes sense as soon as several attributes are defined on arcs.

In this paper, our first objective is to evaluate the tractability of a multigraph representation. We describe this representation in Section 2. Section 3 focuses on the new difficulties for vehicule routing optimization caused by this representation. The paper concludes with the case of a practical DRT system in Section 4.

2 Multigraph representation

Let $G_D = (V_D, A_D)$ be the graph induced by a geographic network. An arc of A_D typically represents a link between two crossroads or a portion of road having consistent characteristics (slope, direction...). G_D 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 $(i, j) \in A_D$ is characterized by R attributes ($R \geq 2$): $d_{D_{ij}}(1), \ldots, d_{D_{ij}}(R)$. Attributes can indifferently represent duration, distance, cost, interest, onerousness, 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_D playing a special part in the problem: vehicle depots, customer locations, origin and destination of transportation requests... Let $V \subset V_D$ 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 from i to j, considering the R criteria $d_D(1), \ldots, d_D(R)$. We introduce the multigraph $G = (V, A)$. For each couple of vertices $(i, j) \in V \times V$ and each path $P^e_{ij} \in \mathcal{P}_{ij}$ ($1 \leq e \leq |\mathcal{P}_{ij}|$), we introduce an arc $(i, j)^e \in A$. The arc $(i, j)^e$ is then characterized by $d^e_{ij}(1), \ldots, d^e_{ij}(R)$.

Note that sets $\mathcal{P}_{ij}$ are possibly of very large size, especially when R is large. However, one can expect to have $R = 2$ or $R = 3$ in most practical cases. Also, attributes time, distance and cost are generally closely correlated, which can drastically limit the number of Pareto optimal paths. Finally, one could imagine proposing to users a heuristic preselection of a subset of paths from $\mathcal{P}_{ij}$ in case it is too large (and depending of the context). In the remaining, we assume that such a multigraph $G = (V, A)$ makes part of input data.

3 Route optimization in a multigraph

Vehicle routing problems generally address three types of decision:

- 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 but can be solved efficiently through dynamic programming. Besides the FSASP tackled in Section 3.1, this Section focuses on the impact of the multigraph representation on the standard solution schemes of vehicle routing problems. The impact on resolution algorithms is briefly evoked in Section 3.2. The particular case of the insertion operator in a sequence, widely used in neighborhood definitions, is studied in Section 3.3.

3.1 Fixed Sequence Arc Selection Problem

The theorem 1 proves the NP-hardness of **FSASP** as a reduction (Figure 2) of a Multidimensional Multiple Choice Knapsack Problem (**MMCKP**), a NP-hard problem as an extension of the Knapsack Problem (Martello and Toth, 1990). Let us define the

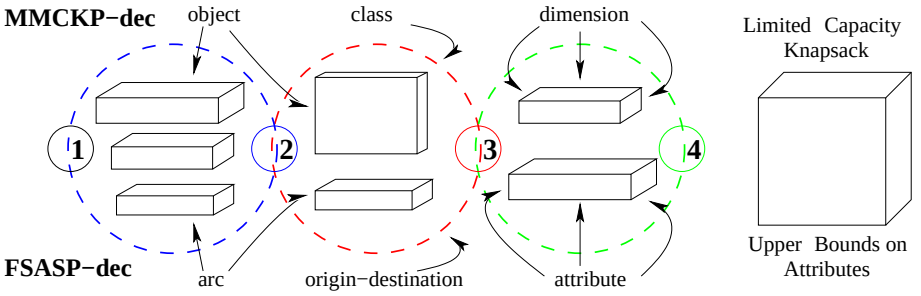

Figure 2: Reduction from MMCKP to FSASP

FSASP more precisely. Let $G = (V, A)$ be a multigraph and R a set of attributes as defined in Section 2. Attribute 0 corresponds to the objective function. An upper bound Q^r is defined for $1 \leq R$. A sequence $(i_0, \ldots, i_N)$ is introduced. 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 the upper bounds are satisfied.

Theorem 1 *The FSASP is NP-hard.*

PROOF. Let us define FSASP-dec the decision problem version of the FSASP. FSASP-dec trivially belongs to the NP class, since it is easy to find a polynomial-length certificate that can be checked in a polynomial time. The NP-completeness is shown with a polynomial time reduction from the decisionnal version of the Multidimensional Multiple Choice Knapsack Problem (MMCKP-dec). The MMCKP-dec is defined as follow, to select in a set of N class j of C_j objects with D dimensions, a set of objects u such that the sum of their value (w_u^d) on each dimension d does not exceed an upper bound W^d. Let us define the following instance of FSASP-dec such that an object is an arc, a class is a destination vertex and a dimension is an attribute. $V = \{i_0, i_1, \ldots, i_N\}$; $R = D$; upper bound of the attribute $Q^r = W^r$ $(r = 1, \ldots, R)$; for the e^{th} object u of class C_j $(1 \leq e \leq |C_j|)$, we associate the ingoing arc j^e of vertex j with attributes $d_j^e(r) = w_u^r$. An optimal arcs selection of FSASP-dec corresponds to an optimal objects selection

Though NP-hard, the problem can be solved rather efficiently. It can be addressed as a Shortest Path Problem with Resource Constraints (SPPRC), generally NP-hard. Resources correspond to the level of consumption on resources. $d_{ij}^{e}(r)$ indicates the level of consumption of resource r when arc $(i, j)^e$ is traversed.

If we assume that resources have non-decreasing extension functions, it can be solved with dynamic programming (Desrosiers *et al.*. 1995). The algorithm (Figure 3) 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 (by the extension functions) 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 have to consider every outgoing arc when extending labels.

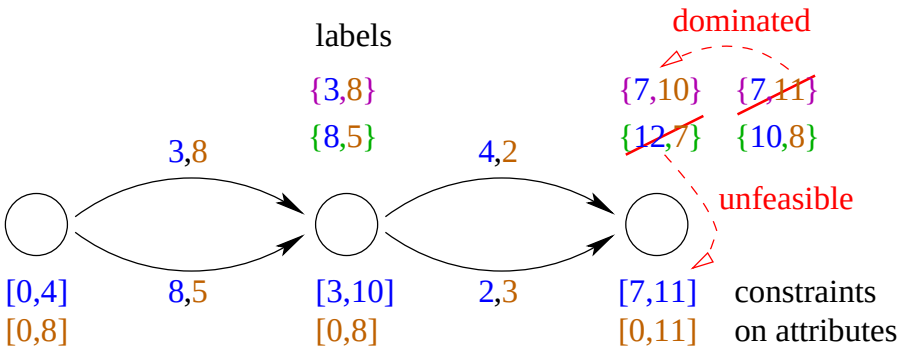

Figure 3: Dynamic programming algorithm for FSASP

When searching for the optimal arc set to use 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 limited number of vertices) , which helps it finding optimal solutions efficiently (Irnich and Desaulniers, 2004).

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 is worse than the incumbent solution. The algorithm then stops. Several metaheuristics mechanisms can be added to avoid being trapped

into local optima. But, in every case, 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 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. In this case, if a resource is very restrictive (like tight time windows), the feasible neighbourhood can be small enough to be exhaustively explored.

With regards to exact methods, multigraph increases the size of data and so the combinatorial aspect of the problem. 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 bounding rule 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... Actually, a noticeable difference relates to the branching scheme. Usually, branching decisions enforce or forbid the use of an arc. With the multigraph representation, this can be rather inefficient, as forbidding an arc is not as strong as in the simple graph case.

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 consider the acyclic multigraph $G_1 = (V_1, A_1)$ where V_1 is made of the vertices 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 mentioned before by adding a resource R_s implying the insertion of exactly one virtual vertex. Finding the best insertion position for s is then equivalent to find the shortest path in G_1 with respect to all resource constraints.

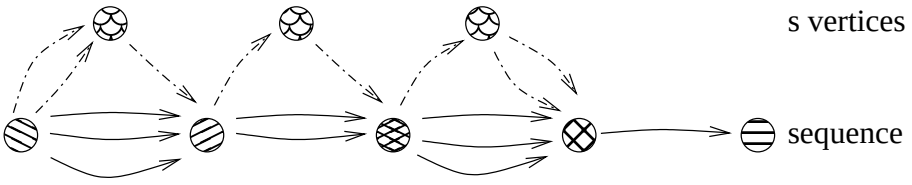

Figure 4: Insertion in a sequence – multigraph construction

The label associated with a partial path is defined by a level of consumption for each resource, a cost, a destination vertex plus the special resource R_s. R_s rules of consumption and violation are summarized in Table 3. This resource is initialized, at the depot, with a value 0. Ingoing arcs on virtual vertices consume 1 unit of the resource, others arcs consume 0. An upper bound upon the consumption of the resource is defined with a value 1 for every virtual vertex; labels extended to the final depot are constrained to have a value 1. The extension function of this resource is trivially non-decreasing. We present further a more detailed algorithm for the case of a request insertion.

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

Table 1: Insertion resource extension function

4 Example of a DRT system in the Doubs Central area

4.1 DRT description

The motivation of this study stems from the development of a Demand Responsive Transport (DRT) system in the Doubs Central area (France). A DRT system is a flexible transport system intented to carry out transport requests via a fleet of vehicles under feasibility and operational constraints. Contrary to a traditional public transport system, the routes are determined for each time period (a day in our case), according to the requests. The key issue for such systems is to find optimized operational solutions taking account of the possibly contradictory objectives of the partners involved :

- for the Transport Organizing Authorities (TOA), rationalize the transport offer and make the service visible and attractive;
- for the conveyors (Local Taxi companies in the central Doubs case), make the service profitable;
- for the possible subcontractors (hauliers), conquer new markets;
- for the associations of users, improve the quality of life and the access to the facilities.

In the DRT system considered here, each user issues a request defined by a pick-up point (departure), a drop-off point (arrival), a given 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 this latest drop-off date. As we will specify it later, these constraints can be expressed using traditional time windows. Local taxi companies carry out the service, so the fleet is

heterogeneous, limited in quantity and with multiple depots. The payment is proportional to the distance plus a fixed cost for the use of each vehicle. Distances, both in time and in kilometers, are known for the road network. Our objective is to find the less costly transportation plan for the authorities and, then, to minimize the time spend in transport by the users.

In this case, the interest to consider alternative paths is to propose less expensive paths (avoiding tolls for example), but proposing an equivalent quality of service.

DRT are well studied systems. The underlying vehicle routing problem is generally identified under the name of *Dial-a-Ride Problem* (DARP). The DARP is a particular case of *Pickup & Delivery Problem* which consists in transporting goods from points of collecting, to points of delivery; the distinction comes from the quality of service having to be integrated when people are carried. Most of the work on the DARP has realistic application. The reader may find a recent state-of-the-art on this subject in (Cordeau and Laporte, 2003). The DARP met in Doubs Central can be modelled (refer to Section 4.2) as a *Pickup & Delivery Problem with Time Windows* (PDPTW), a very well studied problem. Among the different methodologies proposed for its resolution, one can cite tabu search and simulated annealing for heuristics (Li and Lim, 2001) and column generation methods (Savelsbergh and Sol, 1995, Sigurd *et al.*, 2004) for exacts methods. Toth and Vigo (2002) and Crainic and Laporte (1998) present more general information on *Pickup & Delivery Problem* and other vehicle routing problems. The originality of our problem compared to the previous ones lies in the multigraph feature. Especially, as highlighted before, this model introduces another decision level, the sequence arc selection.

4.2 Problem Formulation

The problem is to serve a set $\mathcal{R}$ of requests with an heterogeneous fleet of K vehicles of limited capacity C_k, based in various depots and with a fixed cost pc_k. Let $G = (V, A)$ be a network, where V is the set of nodes including two nodes for each request, consisting of two services r^+ for the pick-up and r^- for the drop off, and two nodes for each vehicle , one for the starting depot and one for the arrival depot (respectively noted o_k and m_k). The request r brings together l_{r^+} passengers. We define also the value $l_{r^-} = -l_{r^+}$. A service i (r^+ or r^-) has a non-negative duration s_i. An arc $(i, j)^e \in A$ is a path (in the geographical network) linking i to j. A cost $d^e_{ij}(2)$, a load $d^e_{ij}(1) = l_j$ and a duration $d^e_{ij}(0)$ are associated with each arc $(i, j)^e$. The arcs corresponding to the shortest time paths in the geographical network are identified by the value $e = 0$.

For each request $r \in \mathcal{R}$ we note B_{r^-} the latest drop-off date and δ_r the maximal gap between B_{r^-} and the effective pick-up date; δ_r is proportionnal to the shortest time path ($d^0_{r^+r^-}(0)$). Equations (1)-(3) turn these constraints into time windows constraints, where A_i is the earliest starting date for the service i and B_i the latest starting date. This formulation splits up the gap constraint (over the two services of a request) into two independent constraints on each service:

$$B_{r^+} = B_{r^-} - d^0_{r^+r^-}(0) - s_{r^+} \quad (1)$$

$$A_{r^+} = B_{r^-} - \delta_r - s_{r^+} \quad (2)$$

$$A_{r^-} = A_{r^+} + s_{r^+} + d^0_{r^+r^-}(0) \quad (3)$$

The part of a solution relative to a single vehicle is called a *r*oute. 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 have to assign one vehicle to each request, to sequence the services for each vehicle and to fix service starting dates T_i, the latter correspondig to arc selection.

To simplify the model, we introduce decision variables L_i that correspond to the number of passengers in the vehicle used, after the service i is done. The binary decision variables x^k_{ije} state the selection of arc $(i, j)^e$ by the vehicle k. The objective function (4) first minimizes the total cost for the authorities. A second hierarchical objective function is added to take the customer quality of service into consideration and to minimize the sum of gaps between latest drop-off dates and effective pick-up dates. To ensure the model coherence, some data are set as follows : $d^e_{m_k o_k}(2) = d^e_{m_k o_k}(1) = d^e_{i m_k}(1) = d^e_{o_k i}(1) = 0$; $A_{o_k} = A_{m_k} = -\infty$ and $B_{o_k} = B_{m_k} = \infty$.

$$\min \text{Lex}\Big(\sum_{k\in K}\sum_{(i,j)^e\in A} x^k_{ije} d^e_{ij}(2) + \sum_{k\in K}\sum_{(o_k,i)^e\in A} pc_k x^k_{o_k ie}, \sum_{i\in N} B_{i^-} - T_{i^+}\Big) \tag{4}$$

subject to

$$\sum_{k\in K}\sum_{(i^+,j)^e\in A} x^k_{i^+ je} = 1 \qquad \forall i \in \mathcal{R}, \tag{5}$$

$$\sum_{(i^+,j)^e\in A} x^k_{i^+ je} - \sum_{(j,i^-)^e\in A} x^k_{ji^- e} = 0 \qquad \forall k \in K, \forall i \in \mathcal{R}, \tag{6}$$

$$\sum_{(i,j)^e\in A} x^k_{ije} - \sum_{(j,i)^e\in A} x^k_{jie} = 0 \qquad \forall k \in K, \forall j \in V, \tag{7}$$

$$x^k_{ije}\left(T_i + d^e_{ij}(0) + s_i - T_j\right) \le 0 \quad \forall k \in K, (i, j)^e \in A, \tag{8}$$

$$A_i \le T_i \le B_i \qquad \forall k \in K, i \in V, \tag{9}$$

$$T_{i^+} + d^0_{i^+ i^-}(0) + s_i \le T_{i^-} \qquad \forall k \in K, i \in \mathcal{R}, \tag{10}$$

$$x^k_{ije}\left(L_i + d^e_{ij}(1) - L_j\right) = 0 \quad \forall k \in K, (i, j)^e \in A, \tag{11}$$

$$x^k_{ije} L_{i^+} \le C_k \qquad \forall k \in K, i \in \mathcal{R}, \tag{12}$$

$$x^k_{ije} L_{i^-} \le C_k - d^e_{ij}(1) \qquad \forall k \in K, i \in \mathcal{R}, \tag{13}$$

$$x^k_{ije} \in \{0, 1\} \quad \forall k \in K, (i, j)^e \in A. \tag{14}$$

4.3 Insertion heuristic

We propose a three steps algorithm.

1. The first step is a greedy insertion procedure which aims at constructing an arc sequence per vehicle (without time-stamping) satisfying all requests.
2. Local search, based on removals and insertions, is then used to improve the set of sequences.
3. The arc sequences are scheduled (time-stamped) in the last step.

4.3.1 The three step approach

Greedy insertion procedure Empty sequences are initially associated with each vehicle. The requests are then inserted one after another into sequences, according to a predetermined order. The insertion generating the smallest increase in cost, is selected. Insertion is "greedy" in the sense that the relative order of the yet inserted requests is preserved. On the other hand, the arcs 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 FSASP with 2 attributes (time and cost)(see Section 3.1). The model and the method proposed to solve it are detailed in the following subsection. Let us specify that in the usual case where only one arc is considered between two stops (simple graph), the problem is polynomial.

In order to improve the effectiveness of this phase, the algorithm is applied twice. The first time, requests are treated in 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. 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 the first inserted. The best of the two solutions obtained is preserved.

Local search (descent method) Once a first set of sequences obtained, the principle of the second step is to remove certain requests of the solution, then to re-insert them at lower cost (with the insertion procedure described above). The new solution is thus, either identical to or more advantageous than the previous one. This treatment is applied to the requests in the initial order of insertion which proved empiricaly to be most powerful. This operation is repeated until no more improvement of the solution during a complete cycle is noted.

Time-stamping (scheduling) The solution obtained with our algorithm produces feasible arcs sequences with unfixed starting dates of service. These dates have no impact on the first hierarchical level of the solution cost. To schedule a sequence S, we optimize the second hierarchical level of the objective function which is to 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 (15) since the latest drop-off times are constant :

$$\max \sum_{r \in S} T_{r^+} \tag{15}$$

The recursion procedure described in formula (16), computes latest feasible pick-up dates that characterize the *Latest Scheduling Solutions*. Theorem 2 proves the optimality of these solutions dates for the secondary criteria.

$$B_i := \min \left\{ B_i, B_{i+1} - d^0_{i,i+1}(0) - s_i \right\} \quad \forall i = |2S| - 1, \ldots, 1 \tag{16}$$

Lemma 1 *Let $\mathcal{T}'$ a Latest Scheduling Solution of a service sequence S with service date B'_i. Let $\mathcal{T}$ a solution such that i exists with $T_i > B'_i$. $\mathcal{T}$ is unfeasible.*

PROOF. If $B_i = B'_i$ then $\mathcal{T}$ is unfeasible else $B_i > B'_i$. In that case $B'_i = B'_{i+1} - d^0_{i,i+1}(0) - s_i$, so $T_i > B'_{i+1} - d^0_{i,i+1}(0) - s_i$. However $T_{i+1} > T_i + s_i + d^0_{i,i+1}(0)$, that

implies $Ti + 1 > B'_{i+1}$. This process is stopped either by a relation $B_j = B'_j$ and $\mathcal{T}$ is unfeasible, or by the end of the sequence with the equation $T_{2S-1} > B'_{2S-1}$. However $B'_{2S-1} = B_{2S-1}$ by construction, which proves the lemma. □

Theorem 2 *Let S be a service sequence such that to each service i a time window constraint $[A_i, B_i]$, a duration s_i and a realization date T_i are associated. The travel time between two successive services is noted $d^0_{i,i+1}(0)$. If S is feasible then Latest Scheduling Solutions are feasible and maximize the function $\sum_{r \in S} T_i$.*

PROOF. If S is feasible then a solution $\mathcal{T}^0$ exists such that for each i, $A_i \leq T^0_i \leq B_i$. Let $\mathcal{T}'$ be the latest scheduling dates T'_i. $\mathcal{T}'$ is unfeasible if and only if $T'_u < A_u$ for one or more u ($T'_i \leq B_i \, \forall i$, by construction). In this case, $T^0_u > T'_u$ which proves the feasibility of $\mathcal{T}'$ by lemma 1.

Let $\mathcal{T}^1$ a feasible solution better than $\mathcal{T}'$. u exists such that $T^1_u > T'_u$. Lemma 1 proves the no-existence of a such $\mathcal{T}^1$, and then the optimality of $\mathcal{T}'$. □

The drop-off dates (T_{r^-}) from the following rules and equation (17) :

- 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 passengers concerned are dropped off.
- A vehicle leaves a stop immediately after the last pick-up.

$$\forall i \text{ a drop-off}, T_i = T_{i-1} + s_{i-1} + d^0_{i-1,i}(0) \tag{17}$$

Solutions respecting these operationals 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 to the case of a request r of the approach described in Section 3.3. Let us consider the acyclic multigraph $G_1 = (V_1, A_1)$ where vertices are depots, pick-ups and drop-offs of the sequence plus one vertex for r^+ and r^- at each insertion position. A contains all the arcs respecting the sequence order (Figure 5). Initial constraints or constraints induced by the sequence can immediately reduce the number of insertion position.

Finding the best insertion position for r is equivalent to find the shortest path (in cost), 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_j associated with a partial path is defined with a level of consumption for each resource (time T^{lb_j} and load L^{lb_j}), a cost C^{lb_j}, and a destination vertex j, plus the resource request R^{lb_j} for the new request. The rules of consumption and violation of these resources are summarized in table 2 and 3. The R^{lb_j} starting level at the depot is set to 0.

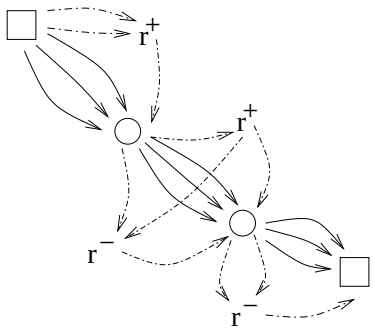

Figure 5: Request insertion in a sequence – multigraph construction

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

Table 2: Resource extension functions

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

Table 3: R^{lb_j} Request resource extension function

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

The dominance rule works as follow. lb_j dominates lb'_j if equations (18) are valid.

$$C^{lb_j} \leq C^{lb'_j}; T^{lb} \leq T^{lb'_j}; L^{lb_j} \leq L^{lb'_j}; R^{lb_j} \geq R^{lb'_j} \tag{18}$$

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

When adding the request load to the maximal load in the sequence do 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 windows constraints. Time windows can be readjusted, preserving all feasible solutions, with a recursive algorithm based on equations (19) and (20). This update is made traversing the sequence twice (one time in each direction) in $O(|2S|)$ where $|S|$ is the number of requests in the sequence S. i is the vertex at the i^{th} position in S.

$$A_{i+1} := \max\left\{A_{i+1}, A_i + s_i + d^0_{i,i+1}(0)\right\} \quad \forall i = 1, \ldots, |2S| - 1 \tag{19}$$

$$B_i := \min\left\{B_i, B_{i+1} - d^0_{i,i+1}(0) - s_i\right\} \quad \forall i = |2S| - 1, \ldots, 1 \tag{20}$$

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 correspond to our situation, even with a simple graph representation.

Thus, we generated our own benchmark from geographical data of Doubs Central (IGN[1] maps) and estimated flows of population. We created three sets of 30 instances with 10, 30 and 90 requests. The fleet is heterogeneous and corresponds to the taxi companies. The maximal gap between the latest drop-off date and the effective pick-up date is two times longer than the shortest time path (in the geographical network). The multigraph is two to three times denser than the simple graph. The variations between arcs connecting the same vertices can go up to 30% of profit in cost for 30% of additional travel time compared to the shortest time path.

The results obtained with the algorithm presented in Section 4.3 (MULTI), are compared in Table 4 with those obtained by considering only the shortest time paths (SIMPLE) with the similar algorithm proposed in Garaix et al., 2005. The number of requests and the algorithm used can respectively be found in the first and second columns. The three following columns indicate the mean cost, number of vehicles used and computing times for the 30 instances of each benchmark. The tests were carried out with a 600MHz AMD Duron and 128 Mo RAM.

MULTI obtains the least expensive solutions on average (2,7%, 3,3% and 3,5% for the instances with 10, 30 and 90 requests), with neither more vehicles used nor more computing time. However results are better with the simple graph, for ten instances.

[1] Institut Gégraphique National

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 multigraph and simple graph

5 Conclusion

In this article, we investigate the interest and the tractability of the use of a multigraph representation for solving vehicle routing problems when arcs of the geographical network are characterized with several attributes.

In a first step, we make clear that this representation renders the problem complex even when the vehicle assignment and sequencing decisions are fixed, i.e., when the problem is reduced to scheduling the services. 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, an emphasis is made on the most shared tool used for routing: the insertion operator.

An algorithm is derived in the case of a DRT system developped in the Doubs Central area in France. 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 on the introduction of the article.

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 one shot every neighbor solution for the insertion operator is indeed a good starting point for proposing equivalent algorithms for other types of operators. Anyway, it has be shown that any type of operator can be used, if one accepts that the evaluation of a neighbor 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, as explained previously.

References

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

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

Desrosiers J., Dumas Y., Solomon M.M., and Soumis F. 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.

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. In *SAGEO'2005*, 2005.

S. Irnich and G. Desaulniers. Shortest path problems with resource constraints. Technical report, les cahiers du GERAD G-2004-11, CRT, Montréal, 2004.

H. Li and Lim A. A metaheuristic for the pickup and delivery problem with time windows. In *ICTAI 2001*, 2001.

Martello S. and Toth P. *Knapsack Problems: Algorithms and Computer Implementations*. Wiley, 1990.

M W P. Savelsbergh and M. Sol. The general pickup and delivery problem. *Transportation Science*, 29:107–121, 1995.

M. Sigurd, D. Pising, and Sig M. The pickup and delivery problem with time windows and precedences. Technical report, Dpt. of Computer Science, University of Copenhagen, 2004.

P. Toth and D. Vigo. *The vehicle routing problem*. Society for industrial and Applied Mathematics, Philadelphia, PA, USA, 2002.