Optimization of occupancy rate in dial-a-ride problems via linear fractional column generation

Thierry Garaix, Christian Artigues, Dominique Feillet, Didier Josselin

Optimization of occupancy rate in dial-a-ride problems via linear fractional column generation

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

[a] *Politecnico di Torino, DAUIN, Torino, Italy.*
[b] *CNRS; LAAS; 7 avenue du colonel Roche, F-31077 Toulouse, France.*
[c] *Université de Toulouse; UPS, INSA, INP, ISAE; LAAS; F-31077 Toulouse, France.*
[d] *Ecole des Mines de Saint-Etienne, CMP Georges Charpak, F-13541 Gardanne, France.*
[e] *Université d'Avignon et des Pays de Vaucluse, UMR ESPACE 6012 CNRS, F-84911 Avignon, France.*

Abstract

In this paper, we consider a dial-a-ride problem where the objective is to maximize the passenger occupancy rate. The problem arises from an on-demand transportation system developed in a rural zone in France, where the objective of encouraging people meeting is pursued. We address the solution of the problem with a column generation approach, applied to a set partitioning formulation where the objective function is fractional. Based on the literature on linear fractional programming, two methods are developed to deal with this fractional objective. Experiments permit to compare these two approaches and to evaluate the impact of the new objective compared to a standard min-cost or min-time optimization.

Keywords: vehicle routing, on-demand transportation, linear fractional program, dial-a-ride problem, column generation.

1. Introduction

The motivation for this study stems from a multidisciplinary research project dealing with on-demand transportation (ODT) systems and their adequacy with new mobility practices (Josselin and Genre-Grandpierre, 2005). In this context, an ODT reservation software was developed for the Doubs Central area, a rural zone in France. ODT systems are flexible passenger transportation systems, where, contrarily to traditional transportation systems, routes are determined on a daily basis (or, at least, for short time periods), according to passenger requests. Many objectives (conflicting or not) can be pursued when developing

Email address: garaix@emse.fr (Thierry Garaix)

such systems: rationalize and make the service attractive (transportation organizing authorities), maximize profits and conquer new markets (carriers, *e.g.* taxis), improve quality of life and access to facilities (passengers).

In this paper, we address a new objective, compared to the literature on ODT systems: the maximization of the passenger occupancy rate. This rate is defined as the sum of the passenger travel times divided by the total travel time of vehicles. The aim is to encourage passenger meeting during the transportation and thus develop or maintain social cohesion. As a side effect, it could also induce passenger gathering for future requests. The relevance of this objective emerged from discussion with transportation organizing authorities from the Doubs Central.

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 the Pickup & Delivery Problem with Time Windows (PDPTW) where people are transported instead of goods. As a consequence, new constraints or objectives related to quality of service have to be addressed. Most of the work on the DARP being issued from real-life applications, a large variety of slightly different problems have been investigated. The interested reader may find detailed state-of-the-art reviews in Cordeau and Laporte (2003b) and Cordeau and Laporte (2007). Berbeglia et al. (2007), Parragh et al. (2008a) and Parragh et al. (2008b) present extensive reviews on Pickup & Delivery Problems. More general information on these categories of problems can also be found in Golden et al. (2008), Desaulniers et al. (2002) or Crainic and Laporte (1998).

Column generation (actually, branch-and-price) has emerged as one of the most powerful exact solution approaches when dealing with these categories of problems (Boschetti et al., 2008; Ropke and Cordeau, 2009; Parragh et al., 2009). It involves a set partitioning formulation (master problem) where columns are added dynamically with the help of a subproblem. The method is specially efficient when the set of routes remains relatively limited, that is, when the possibilities of combining customers into routes is strongly limited by time and capacity restrictions. Seeing that our application case shows such properties, we propose to address the solution of the DARP defined from the Doubs Central ODT case with this type of approach. The main difficulty lies in the occupancy rate objective that implies a linear fractional objective function in the set partitioning formulation. Based on the literature on linear fractional programming, we developed two methods to deal with this fractional objective.

The contributions of this work are twofold. First, we investigate a new quality-of-service criterion in ODT systems. We propose an exact solution approach and numerical experiments in order to evaluate how tractable is this new criterion and what is its impact on solutions. Secondly, we show how fractional objectives can be handled in column generation procedures. Again, the tractability of the approach is evaluated computationally.

After having presented the ODT system in Section 2 and the general column generation methodology in Section 3, the two solution methods are described in Section 4. Experiments permit, in Section 5, to compare these two approaches

and to evaluate the impact of the new objective against a standard min-cost or min-time optimization.

2. On-Demand Transportation system description

The problem is to serve a set $\mathcal{R}$ of requests with a heterogeneous fleet. A request $r \in \mathcal{R}$ is defined by a pick-up point r^+, a drop-off point r^-, a positive number of passengers F_r, either a latest drop-off time $T^{sup}_{r^-}$ (*outbound* requests) or an earliest pick-up time $T^{inf}_{r^+}$ (*inbound* requests), and a maximal ride time g_r. Outbound requests typically correspond to travels with a target time on arrival, e.g., morning travels to work, travels for surgeries, travels to a train station, etc. Contrariwise, inbound requests represent travels with an availability time (evening travels back home, etc.). The sets of pick-up and drop-off points are respectively denoted $\mathcal{R}^+$ and $\mathcal{R}^-$. Pick-ups and drop-offs are called services. A service $i \in \mathcal{R}^+ \cup \mathcal{R}^-$ has a nonnegative duration S_i.

The ride time limit aims at ensuring a certain quality of service to customers. As shown by Figure 1, it does not only include the time spent by a passenger in the vehicle but also the waiting time either between the completion of the drop-off service and the latest drop-off time $T^{sup}_{r^-}$ (outbound request) or between time $T^{inf}_{r^+}$ and the beginning of the pick-up service (inbound request). It is important to notice that this definition differs from the definition that can be found in several papers (e.g.,Cordeau and Laporte (2003a); Cordeau (2006)). We indeed consider here that customers have no interest in arriving in advance (outbound requests) or depart later (inbound requests).

In practice, g_r is proportional to the min-time path connecting r^+ to r^-. As will be shown below, time window constraints for services can be derived from these data.

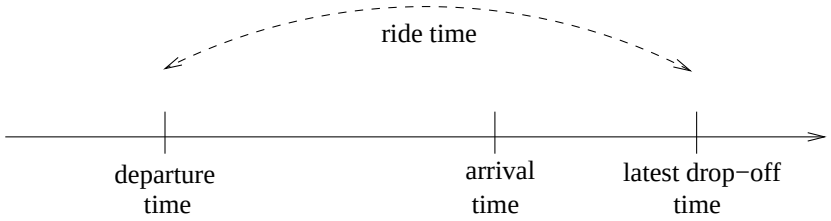

Figure 1: Ride time for an outbound request

The fleet of vehicles is composed of different types of vehicles. $\mathcal{VT}$ represents the set of vehicle types. The set of identical vehicles of type $k \in \mathcal{VT}$ is K^k. A vehicle type $k \in \mathcal{VT}$ is characterized by a capacity F_k and by starting and arrival depots k^+ and k^-. The number of vehicles of type k is $\left|K^k\right|$. The sets of starting and arrival depots are denoted $\mathcal{K}^+$ and $\mathcal{K}^-$, respectively.

Let $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ be a directed graph. $\mathcal{V} = \mathcal{R}^+ \cup \mathcal{R}^- \cup \mathcal{K}^+ \cup \mathcal{K}^-$. An arc $(i, j) \in \mathcal{A}$ is a road-path linking node $i \in \mathcal{V}$ to node $j \in \mathcal{V}$. A distance D_{ij}

and a duration T_{ij} are associated with arc (i, j). For sake of simplifying the notation, we consider that these values are independent of the vehicles. The problem is to find routes for the vehicles such that every request is satisfied, the passenger occupancy rate is maximized and every constraint defined for the vehicles (depots, capacities) and the passengers (drop-off or pick-up time, maximal ride time) are satisfied.

We define the occupancy rate as the ratio between the time spent by passengers in vehicles while these vehicles are driving and the total driving time. Waiting times are not included in our expression. With decision variables $\delta_{ij} = 1$ if arc (i, j) is in the solution, 0 otherwise, and f_{ij} equal to the number of passengers traversing arc (i, j), the occupancy rate can be expressed as:

$$\sum_{(i,j)\in\mathcal{A}} f_{ij}T_{ij} / \sum_{(i,j)\in\mathcal{A}} T_{ij}\delta_{ij} \tag{1}$$

As explained above, earliest pick-up or latest drop-off time constraints combined with the maximal ride time constraints can be equivalently expressed as time windows $[T_i^{inf}, T_i^{sup}]$ for $i \in \mathcal{R}^+ \cup \mathcal{R}^-$. In case of *inbound* requests, these time windows are computed recursively as follows:

$$T_{r^-}^{inf} = T_{r^+}^{inf} + S_{r^+} + T_{r^+r^-} \tag{2}$$
$$T_{r^-}^{sup} = T_{r^+}^{inf} + g_r + S_{r^+} \tag{3}$$
$$T_{r^+}^{sup} = T_{r^-}^{sup} - T_{r^+r^-} - S_{r^+} \tag{4}$$

Equivalent equations allow to compute time windows for *outbound* requests.

Similarly, bounds $[F_{ik}^{inf}, F_{ik}^{sup}]$ on the flow of passengers in a vehicle of type k through $i \in \mathcal{R}^+ \cup \mathcal{R}^- \cup \mathcal{K}^-$, can replace capacity constraints. Let us note $F_{r^-} = -F_{r^+} = -F_r$ for every request $r \in \mathcal{R}$ and $F_{k^+} = F_{k^-} = 0$ for every depot k^+ and k^-, with $k \in \mathcal{VT}$. For a vehicle type k and a node $i \in \mathcal{V}$:

$F_{ik}^{inf} = F_i$ if $i \in \mathcal{R}^+$ — $F_{ik}^{sup} = F_k$ if $i \in \mathcal{R}^+$
$F_{ik}^{inf} = 0$ if $i \in \mathcal{R}^-$ — $F_{ik}^{sup} = F_k + F_i$ if $i \in \mathcal{R}^-$
$F_{ik}^{inf} = 0$ if $i \in \mathcal{K}^+ \cup \mathcal{K}^-$ — $F_{ik}^{sup} = 0$ if $i \in \mathcal{K}^+ \cup \mathcal{K}^-$

3. Column generation approach

In this section, we present a standard column generation scheme as we would have applied it with a classical linear objective function, say minimizing the total travel cost. Similar frameworks can be found in the literature (Dumas et al., 1989; Savelsbergh and Sol, 1998; Ropke and Cordeau, 2009). Our objective in this section is threefold: to remind of the column generation method, to describe the backbone of the two new algorithms presented subsequently in Section 4 and to set a comparison basis for the experiments of Section 5. The reader is referred to Desaulniers et al. (2005) for more details on column generation techniques.

3.1. Framework

Let Ω_k be the set of possible time-stamped routes for a vehicle of type $k \in \mathcal{VT}$, carrying out at most once every transportation request, satisfying time window and capacity constraints. Let $\Omega = \bigcup_{k \in \mathcal{VT}} \Omega_k = \{\omega_1, \ldots, \omega_{|\Omega|}\}$ be the complete set of possible routes (identical routes assigned to different vehicle types are considered different). Let $b_{kn} = 1$ if route $\omega_n \in \Omega_k$, $b_{kn} = 0$ otherwise. Let $a_{rn} = 1$ if route $\omega_n \in \Omega$ carries out request r, $a_{rn} = 0$ otherwise. Let us assume that a cost C_{ij} is defined on arcs $(i, j) \in \mathcal{A}$ and let c_n be the cost of route $\omega_n \in \Omega$. The DARP can be stated as follows:

$$\min \sum_{\omega_n \in \Omega} c_n \lambda_n \tag{5}$$

subject to

$$\sum_{\omega_n \in \Omega} a_{rn} \lambda_n = 1 \quad \forall r \in \mathcal{R}, \tag{6}$$

$$\sum_{\omega_n \in \Omega} b_{kn} \lambda_n \leq \left|K^k\right| \quad \forall k \in \mathcal{VT}, \tag{7}$$

$$\lambda_n \text{ integer} \quad \forall \omega_n \in \Omega. \tag{8}$$

Decision variables λ_n indicate whether route $\omega_n \in \Omega$ is selected or not in the solution. Objective function (5) minimizes the total travel cost. Constraints (6) ensure that every request is carried out exactly once. Constraints (7) limit the number of vehicles of each type.

Solving the linear relaxation of model (5)-(8) 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 (5)-(8). Column generation is based on two components: a restricted master problem and one or several slave problems. The restricted master problem MP($\tilde{\Omega}$) is obtained from MP by considering only a subset $\tilde{\Omega} \subset \Omega$ of variables. A slave problem aims at adding progressively new potentially good columns to $\tilde{\Omega}$ until an optimality criterion is attained. Here, we introduce a slave problem for every vehicle type.

At every iteration of the algorithm, MP($\tilde{\Omega}$) is solved with the simplex method. A slave problem determines for a specific vehicle type k whether some variables λ_n with $\omega_n \in \Omega_k$ have a negative reduced cost. This condition can easily be stated as:

$$c_n - \sum_{r \in \mathcal{R}} a_{rn} \pi_r - \mu_k < 0, \tag{9}$$

where π_r is the dual variable associated with constraint (6) for request r and μ_k is the dual variable associated with constraint (7) for vehicle type k.

One or several variables with negative reduced cost are then added to $\tilde{\Omega}$ and the algorithm iterates until all slave problems fail to find new routes. At each iteration, only a subset of slave problems are solved and slave problems are not necessarily solved to optimality. The purpose is to find out routes with negative reduced cost and therefore solving is stopped when enough routes are found. A

score measure is computed to determine the order in which slave problems are solved. A score is obtained for a slave problem by combining a random value and the dual value μ_k relative to the considered vehicle type.

Classically, the branching scheme consists in enforcing or forbidding 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.

Further details on this algorithm can be found in Garaix et al. (2010).

3.2. Solution of the slave problems

A slave problem, for a vehicle type $k \in \mathcal{VT}$, can be seen as an *Elementary Shortest Path Problem with Resource Constraints* (ESPPRC). The aim is to find, in graph $\mathcal{G}$, from starting depot k^+ to arrival depot k^-, an elementary path of minimal cost subject to resource constraints. The path has to be elementary in the sense that requests should not be carried out more than once. We propose to solve slave problems with a dynamic programming approach, adapted from Feillet et al. (2004). The principle is to associate with each possible partial path a label and to extend these labels checking resource constraints until feasible paths of negative reduced cost are obtained. Dominance rules are used to compare partial paths arriving at a same node and to discard some of them.

Labels are defined with the following 8 attributes: a pointer to the parent label $fath$, the current ending node en, resources F and T for the load and time consumption, cost C, and the sets of respectively open, closed and time-reachable requests $\mathcal{O}, \mathcal{C}, \mathcal{T}$. A request is said to be open once the pick-up service is carried out and as long as the drop-off service has not been performed. Then, the request is said to be closed. Time-reachable requests are requests that can be reached according to time windows: $r \in \mathcal{T}$ when the successive extensions to r^+ and r^- are valid considering $\left[T^{inf}_{r^+}, T^{sup}_{r^+}\right]$ and $\left[T^{inf}_{r^-}, T^{sup}_{r^-}\right]$. $\mathcal{C}$ and $\mathcal{O}$ resources ensure the elementariness, the pick-up and drop-off matching and the precedence properties.

Extension functions and feasibility conditions when extending labels are described in Table 1 and Table 2. In these tables, the extension of a label L to a label M, through an arc (i, j) is considered (with $i = en^L$ and $j \in \mathcal{R}^+ \cup \mathcal{R}^- \cup \{k^-\}$). In Table 1, remind that $F_{r^-} = -F_{r^+} = -F_r$ for every request $r \in \mathcal{R}$ and $F_{k^-} = 0$. Also, we define $\pi_j = \pi_r$ if $j = r^+$, $\pi_j = 0$ if $j = r^-$ and $\pi_j = \mu_k$ if $j = k^-$. In Table 2 extension functions and feasibility conditions are given according to the nature of j.

In Table 2 notation $TSPTW(en^M, \mathcal{O}^M, k^-)$ represents the problem of finding a path reaching k^- from en^M such that every drop-off point of the requests included in $\mathcal{O}^M$ is visited. The feasibility of $TSPTW(en^M, \mathcal{O}^M, k^-)$ is relatively easy to compute for small sets of open requests, which can be expected in practice because of the tightness of time window and capacity constraints.

The dominance rule works as follows. L dominates L' if the following conditions hold:

resource	value	constraint
$fath^M$	L	$-$
en^M	j	$-$
T^M	$max\left\{T^L + S_i + T_{ij}, T_j^{inf}\right\}$	$\leq T_j^{sup}$
F^M	$F^L + F_j$	$\in \left[F_{jk}^{inf}, F_{jk}^{sup}\right]$
C^M	$C^L + C_{ij} - \pi_j$	*to minimize*

Table 1: Extension functions for resources $fath$, en, T, F and C

	resource	$j = r^+$	$j = r^-$	$j = k^-$
value	$\mathcal{O}^M$	$\mathcal{O}^L \cup \{r\}$	$\mathcal{O}^L \setminus \{r\}$	$\mathcal{O}^L$
	$\mathcal{C}^M$	$\mathcal{C}^L$	$\mathcal{C}^L \cup \{r\}$	$\mathcal{C}^L$
	$\mathcal{T}^M$	see definition above		
constraints		$r \notin \mathcal{C}^L$	$r \in \mathcal{O}^L$	$\mathcal{O}^L = \emptyset$
		$TSPTW(en^M, \mathcal{O}^M, k^-)$ is feasible		

Table 2: Extension functions for resources $\mathcal{O}$, $\mathcal{C}$ and $\mathcal{T}$

$$en^L = en^{L'} \quad (10) \qquad C^L \leq C^{L'} \quad (13)$$

$$T^L \leq T^{L'} \quad (11) \qquad \mathcal{O}_L = \mathcal{O}_{L'} \quad (14)$$

$$F^L \leq F^{L'} \quad (12) \qquad \mathcal{C}_L \cap \mathcal{T}_{L'} \subset \mathcal{C}_{L'} \quad (15)$$

Condition (14) enforces that labels L and L' have the same open requests. According to condition (15), all the requests completed in L are either also completed in L' or not time-reachable for L'. Combined with conditions (10) and (11), it ensures that every request that could be carried out by L' in the future can also be carried out by L. Note that the open request equality constraint (14) implies flow constraint (12), which is only expressed here for sake of clarity. Note also that for sake of computational efficiency condition (15) can be rewritten:

$$(\mathcal{C}_L \cap \mathcal{T}_L) \cap \mathcal{T}_{L'} \subset (\mathcal{C}_{L'} \cap \mathcal{T}_{L'}) \quad (16)$$

thus avoiding spanning the entire set $\mathcal{C}_L$ while comparing labels.

Stronger dominance rules, managing inclusion between open requests sets $\mathcal{O}_L$ and $\mathcal{O}_{L'}$ have been investigated in the literature (Sol, 1994; Ropke and Cordeau, 2009), but were not adapted to our study as they require standard linear cost definitions.

The resource level for the initial label defined at the depot k^+ is set to 0 (or $\emptyset$) for every resource. At each step of the algorithm, a label is selected and extended in every possible direction. The labels waiting for extension are stocked in a priority queue. In order to strengthen the dominance rule impact,

labels are ordered according to time, except at the beginning of the algorithm where min-cost labels are preferred (labels of negative cost being quite easy to find).

4. Occupancy rate criterion integration

In order to introduce the occupancy rate objective in the master problem, we introduce new notation. With every column $\omega_n \in \Omega$ are associated N_n and D_n such that $N_n = \sum_{(i,j)\in\omega_n} f_{ij}^n T_{ij}$ and $D_n = \sum_{(i,j)\in\omega_n} T_{ij}$, where $(i,j) \in \omega_n$ indicates that route ω_n traverses arc (i,j) and f_{ij}^n is the number of passengers in the vehicle at that time.

The problem to solve then becomes:

$$\max \sum_{\omega_n\in\Omega} N_n\lambda_n / \sum_{\omega_n\in\Omega} D_n\lambda_n \tag{17}$$

subject to (6)-(8).

Objective (17) is the transcription with variables λ_n of the occupancy rate objective (1) defined in Section 2. In the remainder, $N(\lambda)$ and $D(\lambda)$ are abbreviated notation for the numerator $\sum_{\omega_n\in\Omega} N_n\lambda_n$ and the denominator $\sum_{\omega_n\in\Omega} D_n\lambda_n$, respectively.

The purpose of this section is to adapt the previous column generation scheme to this linear fractional objective. We propose two approaches for solving the master problem. The first one, called direct approach, uses a change of variables and was first introduced by Charnes and Cooper (1962). The second one, called iterative approach, is based on Dinkelbach's algorithm (Dinkelbach, 1967). In both cases, the original method was not designed to be combined with column generation. As far as we know, though many other and more recent algorithms exist to address fractional objectives, the two approaches cited above are still state-of-the-art for linear fractional objectives (Bazaraa et al., 2006). Furthermore, we did not find any attempt to combine these approaches with column generation, except a quick mention in Jaumard et al. (1991). Also, the only other tentative we found to apply column generation when the objective is fractional was in Gilmore and Gomory (1963), where an adaptation of the simplex method is investigated (see also (Lasdon, 1970)).

4.1. Direct approach

We apply the change of variables $x = 1/D(\lambda)$ and switch the objective to minimization; note that $D(\lambda) > 0$. The master problem can then be written:

$$\min x \sum_{\omega_n\in\Omega} (-N_n\lambda_n) \tag{18}$$

subject to

$$x \sum_{\omega_n \in \Omega} a_{rn} \lambda_n = x \qquad \forall r \in \mathcal{R}, \tag{19}$$

$$x \sum_{\omega_n \in \Omega} b_{kn} \lambda_n \leq x \left| K^k \right| \quad \forall k \in \mathcal{VT}, \tag{20}$$

$$x \sum_{\omega_n \in \Omega} D_n \lambda_n = 1, \tag{21}$$

$$x\lambda_n \geq 0 \quad \forall \omega_n \in \Omega, \tag{22}$$

$$x \geq 0. \tag{23}$$

or equivalently after a new change of variable $\lambda_n \leftarrow x\lambda_n$:

$$\min \sum_{\omega_n \in \Omega} (-N_n \lambda_n) \tag{24}$$

subject to

$$\sum_{\omega_n \in \Omega} a_{rn} \lambda_n - x = 0 \qquad \forall r \in \mathcal{R}, \tag{25}$$

$$\sum_{\omega_n \in \Omega} b_{kn} \lambda_n - x \left| K^k \right| \leq 0 \quad \forall k \in \mathcal{VT}, \tag{26}$$

$$\sum_{\omega_n \in \Omega} D_n \lambda_n = 1, \tag{27}$$

$$\lambda_n \geq 0 \quad \forall \omega_n \in \Omega, \tag{28}$$

$$x \geq 0. \tag{29}$$

Equation (30) gives the reduced cost of a route $\omega_n \in \Omega_k$:

$$-N_n - \sum_{r \in \mathcal{R}} a_{rn} \pi_r - \mu_k - D_n \beta \tag{30}$$

or equivalently,

$$- \sum_{(i,j) \in \omega_n} f_{ij}^n T_{ij} - \sum_{r \in \mathcal{R}} a_{rn} \pi_r - \mu_k - \sum_{(i,j) \in \omega_n} T_{ij} \beta \tag{31}$$

where π_r, μ_k and β are the dual variables associated with constraints (25), (26) and (27), respectively.

For each type of vehicle k, the reduced cost, *i.e.* the function to minimize in the slave problem, expressed with arc variables is then:

$$\sum_{(i,j) \in \mathcal{A}} -[(f_{ij} + \beta) T_{ij} + \pi_j] \delta_{ij} \tag{32}$$

where we remind that f_{ij} is a decision variable indicating the number of passengers in the vehicle on arc (i, j), δ_{ij} decides of the selection of arc (i, j) and

π_j is set to π_r for pick-up points, 0 for drop-off points and μ_k for the arrival depot k^-.

Formula (33) defines the resource extension function for the cost in the slave problem when extending label L to label M through arc (i,j) (with $i = en^L$ and $j \in \mathcal{R}^+ \cup \mathcal{R}^- \cup \{k^-\}$):

$$C^M = C^L - (F^L + \beta)T_{ij} - \pi_j. \tag{33}$$

Other extension functions and the dominance rule are defined exactly as in Section 3. The remaining of the algorithm is also identical to the one presented before (with the slight exception that one has to divide λ_n by x to evaluate the flow on arcs when branching).

4.2. Iterative approach

We now propose to adapt Dinkelbach's algorithm (Dinkelbach, 1967) to the column generation context. Dinkelbach first designed its algorithm to solve nonlinear problems with a fractional objective function, a concave numerator and a (non-zero) convex denominator. The linear relaxation of program (17),(6)-(8) fits these conditions. We call DP this relaxation in the following.

Let introduce the parametric linear program DP(x):

$$\max \sum_{\omega_n \in \Omega} N_n \lambda_n - x \sum_{\omega_n \in \Omega} D_n \lambda_n \tag{34}$$

subject to

$$\sum_{\omega_n \in \Omega} a_{rn} \lambda_n = 1 \quad \forall r \in \mathcal{R}, \tag{35}$$

$$\sum_{\omega_n \in \Omega} b_{kn} \lambda_n \leq \left|K^k\right| \quad \forall k \in \mathcal{VT}, \tag{36}$$

$$\lambda_n \geq 0 \quad \forall \omega_n \in \Omega. \tag{37}$$

Let us note $H(x)$ the value of the optimal solution of DP(x). Dinkelbach (1967) shows that function H is continuous, strictly decreasing, and that the following equivalence holds:

$$\lambda^* \text{ is an optimal solution of DP} \Leftrightarrow H(N(\lambda^*)/D(\lambda^*)) = 0$$

Finding the unique zero x^* of function H thus provides the optimal occupancy rate $N(\lambda^*)/D(\lambda^*)$. Following Newton's method, Dinkelbach proposes to approximate x^* with an iterative algorithm constructing sequence (x^q) defined as:

$$x^{q+1} = N(\lambda^q)/D(\lambda^q),$$

where λ^q is the optimal solution of DP(x^q). The algorithm stops when $H(x^q) < \varepsilon$ ($\varepsilon \geq 0$).

Two ways appear relevant to apply Dinkenlbach's algorithm in our context. A first alternative is to consider Dinkenlbach's algorithm at the higher level and to solve every master problem of the sequence DP(x^q) through column generation. A second alternative is rather to solve the linear relaxation of (17),(6)-(8) with column generation, using Dinkelbach's algorithm for the solution of every restricted master problem. Then, the optimal dual vector of the linear program defined at the last iteration of Dinkelbach's algorithm is used to find new columns. We adopt the second option since we prefer to generate columns with the most meaningful dual variables and we prefer to limit the number of ESPPRC calls.

The (reduced) cost of a route in the slave problem relative to vehicle k and iteration $(q+1)$ of Dinkenlbach's algorithm, expressed in a minimization form to be consistent with the previous sections, is:

$$\sum_{(i,j)\in\mathcal{A}} [(x^q - f_{ij})T_{ij} - \pi_j]\delta_{ij} \tag{38}$$

where π_j is defined as detailed in 4.1.

Formula (39) defines the resource extension function for the cost when extending label L to label M through arc (i,j) (with $i = en^L$ and $j \in \mathcal{R}^+ \cup \mathcal{R}^- \cup \{k^-\}$):

$$C^M = C^L + (x^q - F^L)T_{ij} - \pi_j \tag{39}$$

Other extension functions and the dominance rules are defined as in Section 3. After iteration $q+1$ of Dinkelbach's algorithm, route costs need to be updated in the objective function, according to the new value x^{q+1}. The remaining of the algorithm is identical to the one presented in Section 3.

5. Numerical experiments

Tests are run on a Intel Core 2 with a 2.66 GHz processor, 4G of RAM and a Xubuntu Linux operating system. The restricted master problems are solved by Ilog-CPLEX 12.0. All CPU times are expressed in seconds.

5.1. Instances

In order to evaluate our algorithm and the impact of the occupancy rate criterion on the solutions, we selected three sets of Euclidean benchmarks from the literature (Li and Lim, 2001; Cordeau and Laporte, 2003a; Ropke et al., 2007). Because of the original definition of our problem, that differs from the problem for which these instances were initially designed, these instances were adapted. For each request, we first defined whether the request is an inbound or an outbound request. Then, we set latest drop-off time (outbound request) or pick-up time (inbound request), plus a maximal ride time. Note that maximal ride times (with a different meaning) were defined in these instances; however,

as the objective of maximizing occupancy rate does not make sense without a strong guarantee on the travel times of passengers and as the strength of time constraints is directly related to the maximal ride time value in our case, we do not reproduce as large values for maximal ride times as the one proposed in the benchmarks. The following paragraphs detail how the three sets of instances were adapted.

Tests were also performed on a set of instances based on the road-network of the Doubs Central area for which our on-demand transportation system was developed. These instances are described next.

Li and Lim (2001) built instances to the Pickup & Delivery Problem with Time Windows from Solomon's classical VRPTW instances. We conserved the network but adapted other data. The number of passengers of a request was defined as one tenth of the load associated with the request in the original instance. Service times were set to one unit of time per passenger, *i.e.*, the service time of a request is equal to its number of passengers. The maximal ride time was set to $g_r = 1.5 \times T_{r^+r^-}$ for every request. Every request was considered as *outbound*. If a latest drop-off time $T_{r^-}^{sup}$ was originally defined, we kept it. Otherwise, either we deduced it from the given latest pick-up time (when defined) plus the maximal ride time, or we generated it randomly.

As in Solomon's benchmark, instances are divided in three classes: the 'R' class with a random spatial distribution, the 'C' class with clusters (note that pick-up and drop-off points of a request are not necessarily in the same cluster) and the 'RC' as a mix of the previous classes. Every class contains 10 instances. We considered instances from the 200 and 400 series, including approximatively 100 and 200 requests respectively. In the following, these 60 instances are coined R1_200, C1_200, RC1_200, R1_400, C1_400 and RC1_400.

In Cordeau and Laporte (2003a) and Ropke et al. (2007), the authors define inbound and outbound requests with a common maximal ride time. Contrary to ours, their definition of maximal ride time only includes the time spent in the vehicle by the customers. Furthermore, since the original maximal ride times are large (3 times the longest request), we modified them and set a maximal ride time $g_r = 1.5 \times T_{r^+r^-}$ for every request. Depending on their nature (inbound or outbound), we kept the latest drop-off or pick-up time. Service durations and number of passengers are kept as in the original instances. Also, a single depot with 6-seats vehicles is kept for all the instances. We increased the number of vehicles available until we got feasible problems. We thus obtained two sets of instances.

Ropke et al. (2007) generated two series ('a' and 'b') of instances with a ranging of requests between 16 and 96. In the set 'a', there is only 1 passenger per request while requests of the set 'b' concern up to 6 passengers. Our first set of 24 instances, named *rop_bac*, is derived from the 'b' instances of Ropke et al. (2007).

In the second set of 20 instances, named *cord_tabu* and derived from Cordeau and Laporte (2003a), the number of requests ranges between 24 and 144. The main difference between the two series comes from the number of passengers, that is equal to 1 for every request in *cord_tabu* and ranges between 1 and 6 for

rop_bac.

Finally, 24 instances were constructed using the Doubs Central road-network, completing the size of the benchmark to a total of 128 instances. These instances are divided in series of 25, 50 and 100 requests, with maximal ride times equal to $1.5 \times T_{r^+r^-}$ or $1.3 \times T_{r^+r^-}$. For each pair of these parameters, we generated 4 instances corresponding to different polarities of passenger flows: convergent, multi-convergent, extreme-convergent and random. These instances are split in two sets of 12 instances for the experiments, depending on parameter g_r: DC_1.5 and DC_1.3. The homogeneous fleet is divided in 8 depots with 10 vehicles whose capacities are set to 6.

More details on the characteristics of these instances can be found in Garaix et al. (2007, 2010). Instance files are available upon request.

5.2. Direct versus iterative algorithms

In the following tests, we fixed the error margin for optimal occupancy rates to 10^{-4} and 10^{-1} for total driving durations. The ε parameter of the Iterative algorithm is set to 10^{-4}. All approaches start with an empty initial set of columns. A dummy route satisfying all the requests is introduced, with a large coefficient in the objective function, so that the feasibility of the restricted master problems is ensured. No more than 90 columns are added to the master problem after each call to the slave problem solver.

The 128 instances we built are solved to optimality by the both algorithms within 1 hour.

Table 3 first indicates optimal values and root node deviations from the optimum, for the different series of instances. The two algorithms are then compared through CPU time, number of columns generated and number of nodes explored in the search tree. The averages of these values over the 128 instances are given. In this table, letters 'D' and 'I' represent the Direct and the Iterative algorithms, respectively.

sets of x		root node	opt.	CPU		columns		nodes	
instances	x	deviation		D	I	D	I	D	I
C1_400	10	0.00	0.6600	33	32	10662	10868	1.0	1.0
R1_400	10	0.10	1.0334	13	22	5550	5970	9.0	9.0
RC1_400	10	0.05	0.9193	24	187	3860	4304	291.4	291.4
C1_200	10	0.04	0.7537	2	2	2900	2973	1.4	1.4
R1_200	10	0.12	1.0560	1	3	1454	1711	18.2	18.2
RC1_200	10	0.17	1.0612	1	1	1457	1600	6.8	6.8
rop_bac	24	0.00	2.2681	4	4	2699	2762	1.0	1.0
cord_tabu	20	0.00	0.6321	16	18	5210	5236	1.7	1.7
DC_1.5	12	0.11	1.9132	3	3	1094	1167	2.7	2.3
DC_1.3	12	0.00	1.7153	2	2	814	871	8.4	3.8
average		0.05	1.2926	9.5	23.3	3521.1	3669.7	27.1	26.6

Table 3: Direct vs Iterative algorithms on the 128 instances

A first noticeable result is that root node deviations are very small. As a consequence, a few nodes of the search tree suffice to close the gap. Actually, 97 instances out of 128 are solved at the root node. As expected, the numbers of nodes in the branch-and-price trees are almost identical for both approaches.

Finally, one can observe that the number of columns generated by both algorithms are also very similar, while the variations of solution times are rather difficult to interpret. This issue is developed more deeply with Table 4.

sets of	CPU		Max(CPU)		CPU_MP		CPU_SP		#MP		#SP	
instances	D	I	D	I	D	I	D	I	D	I	D	I
C1_400	33	32	61	61	17	15	14	13	307	514	307	303
R1_400	13	22	25	94	4	11	8	8	180	339	179	198
RC1_400	24	187	182	1814	5	161	14	15	449	1205	417	477
C1_200	2	2	3	3	1	1	1	1	117	210	117	123
R1_200	1	3	4	14	0	1	1	1	91	208	90	122
RC1_200	1	1	1	2	0	0	1	1	65	133	64	75
rop_bac	4	4	39	37	2	2	1	1	144	211	144	149
cord_tabu	16	18	111	152	9	12	6	5	190	282	190	191
DC_1.5	3	3	10	12	0	0	2	2	130	224	212	242
DC_1.3	2	2	6	8	0	0	1	1	110	180	179	202
average	9.5	23.3	47.7	187.9	3.9	17.0	4.5	4.3	173.6	325.3	185.1	200.8

Table 4: Direct vs Iterative algorithms on the 128 instances continued

Table 4 decomposes CPU times for the solution of the different series of instances for the two approaches.

The two first columns ('CPU' and 'Max(CPU)') give the average and maximal computing times over all series of instances. Next columns provide the time spent for the solution of the restricted master problem ('CPU_MP') and for the solution of the slave problem ('CPU_SP'). Also, the numbers of calls to the simplex ('#MP') and to the column generator ('#SP') algorithms are given. Note that one call to the column generator is counted every time the slave problem is called for a given type of vehicle.

One can see that the number of calls to the slave problems and the time spent for the solution of these problems are almost equivalent for the two approaches. On the contrary, the iterative approach necessitates the solution of almost the double of restricted master problems, compared to the direct approach. However, this increase is not visible when concentrating on computing times, except for instances of series R1_400 and RC1_400.

Finally, we can remark from this table that the number of Dinkelbach's algorithm iterations is rather limited at each step of column generation. Indeed, the total number of calls to the (modified) restricted master problem is less than twice the number of calls to the slave problem.

5.3. Occupancy rate maximization versus driving duration minimization

The aim of this section is to evaluate the impact on the solutions of switching from the standard min-driving-duration objective to the objective of maximizing the occupancy rate. Tables 5 and 6 respectively consider the solutions found when applying one of the two criterion for the optimization and provide the values of these solutions against the two criteria. In these tables, occupancy rate maximization is called $max(N/D)$, while minimization of the total driving duration is called $min(D)$. Note that this second objective is exactly the denominator of the occupancy rate. Since the total driving duration can be expressed as a linear objective function, the standard column generation scheme presented in Section 3 was applied.

Table 5 presents the values of the two criteria for solutions whose occupancy rate is maximized. Table 6 presents the values of the two criteria for solutions whose driving duration is minimized. In these tables, the average deviation and the maximal deviation of the criterion that is not optimized against its optimal value is given.

name	N/D	D	%dev(D)	max(%dev(D))	CPU
C1_400	0.6600	8385.9	0.13	0.49	33
R1_400	1.0334	11802.3	0.55	1.16	13
RC1_400	0.9193	12305.0	0.43	0.81	24
C1_200	0.7537	3394.2	0.01	0.09	2
R1_200	1.0560	5519.3	0.34	1.15	1
RC1_200	1.0612	5064.2	0.45	0.98	1
rop_bac	2.2681	699.7	0.02	0.16	4
cord_tabu	0.6321	910.9	0.00	0.00	16
DC_1.5	1.9132	985.9	0.35	1.46	3
DC_1.3	1.7153	1081.6	0.02	0.11	2

Table 5: Occupancy rates and driving durations when the occupancy rate is maximized (max(N/D))

The main conclusion that can be drawn from these tables is that the two objectives are seldom conflicting: whatever the criterion optimized, the values remain very close most of the times. The deterioration of any of the two objectives when the optimization is driven by the other objective never reaches 5%, and is less than 1% on average.

Observing instances DC_1.3 and DC_1.5, one can also notice that the occupancy rate significantly increases inversely to total driving duration when parameter g_r raises. It means that, on those runs, reducing no-load travel times exceeds the increase in individual travel times.

5.4. Impact of instance characteristics

As explained in Subsection 5.1, instances are defined with tight time windows. This definition allows maintaining an acceptable quality of service for

name	N/D	D	%dev(N/D)	max%(dev(N/D))	CPU
C1_400	0.6552	8374.9	0.61	1.87	25
R1_400	1.0239	11737.6	0.88	2.14	16
RC1_400	0.9090	12250.9	1.18	3.68	47
C1_200	0.7537	3393.9	0.00	0.01	3
R1_200	1.0496	5500.7	0.58	2.14	3
RC1_200	1.0462	5041.3	1.40	3.05	2
rop_bac	2.2662	699.5	0.07	0.77	3
cord_tabu	0.6321	910.9	0.00	0.00	15
DC_1.5	1.8978	981.2	0.76	2.32	6
DC_1.3	1.7143	1081.2	0.05	0.46	2

Table 6: Occupancy rates and driving durations when the driving duration is minimized (min(D))

passengers. It is particularly necessary given that the occupancy rate is maximized, which could tend to increase artificially the presence of customers in vehicles. In order to evaluate the performance of our algorithm in the context of larger time windows, we ran additional tests on the 'rop_bac' instances (including less than 100 requests). Results are presented in Tables 7 and 8 when occupancy rate and total driving duration are optimized, respectively. In these tables, time windows are enlarged by increasing maximal ride times: $g_r = \alpha \times T_{r^+r^-}$ with α increasing from 1.5 (original value) up to 6.0.

α	solved	N/D	D	nodes	CPU	max(CPU)
1.5	24	2.2681	699.7	1.0	3.7	38.9
2.0	24	2.4622	673.3	1.1	4.4	44.4
3.0	24	2.9119	676.0	7.6	17.0	225.4
4.0	24	3.2749	693.4	26.2	60.7	1042.2
5.0	24	3.5281	705.1	46.8	69.5	509.0
6.0	24	3.7182	719.4	150.8	466.8	9046.0

Table 7: Impact of time window width when maximizing occupancy rate (N/D) for rop_bac instances

When maximizing the occupancy rate, most of the instances are solved in less than 1 hour except one instance: when $\alpha = 6.0$. Except for this instance, one observes a clear but reasonable increase of computing times with g_r. Regarding the minimization of total driving duration (Table 8), larger time windows also impact computing times, but more slightly. One can conclude that though it penalizes more heavily occupancy rate maximization, enlarging time windows does not invalidate the solution process in both cases.

Note that differences in optimal values for the two criteria and between the two tables are not meaningful for high values of α as these values do not correspond to realistic practical situations.

α	solved	N/D	D	nodes	CPU	max(CPU)
1.5	24	2.2662	699.5	1.0	3.1	33.5
2.0	24	2.4376	669.9	1.4	3.0	27.8
3.0	24	2.6452	634.6	4.2	5.3	49.8
4.0	24	2.8181	609.3	6.1	8.8	61.6
5.0	24	2.9375	589.9	25.3	33.8	482.2
6.0	24	3.0354	576.4	37.7	53.6	486.1

Table 8: Impact of time window width when minimizing total driving duration (D) for rop_bac instances

Finally, in order to evaluate the impact of vehicle capacities on solutions, we carried out additional tests with infinite capacities of vehicles for the Doubs Central instances. These tests were only run with the objective of maximizing occupancy rate. Results show average occupancy rates of 1.7153 and 1.7211 instead of 1.9132 and 1.9360, when maximal ride times are equal to 1.3 and 1.5, respectively. These small increases highlight that the time constraints limit the benefit of having larger vehicles.

6. Conclusion

In this paper, we have investigated the maximization of passenger occupancy rate in dial-a-ride problems. The motivation for this problem and this criterion arose from an on-demand transportation system developed in a rural zone in France, where the objective of encouraging people meeting was pursued. As far as we known, dealing with this objective is new.

We proposed two exact solution algorithms that permitted to solve in limited time (often a few seconds) a large panel of benchmark instances of different types with up to 200 requests. The main difficulty stemmed from the fractional form of the occupancy rate criterion. The two solution algorithms proposed were both based on column generation, namely the Direct Algorithm and the Iterative Algorithm. Experiments showed similar behaviors for the two algorithms. Furthermore, computing times were of the same order of magnitudes as the ones obtained when simply optimizing a standard linear objective function. At this stage, one can conclude that introducing fractional objectives is not necessarily detrimental to the solution of large scale integer programs with column generation, both from theoretical and computational points of views. Investigating this issue for different forms of integer programs, *e.g.* cases where the slave problem is much faster, would certainly deserve future researches. Further works evaluating other ways of combining fractional objectives and column generation, following for example Gilmore and Gomory (1963), would also be interesting.

Another relevant phenomenon shown by the experiments was that optimizing the occupancy rate or the driving time provided very similar solutions in terms of quality for these two criteria. One can conclude that the two objectives are

very similar (as can intuitively be guessed) and that guiding the optimization with the simple standard linear objective should be satisfactory even if solutions optimizing occupancy are expected. One could thus avoid relatively complex developments or use standard softwares that would not be able to manage a fractional objective.

M.S. Bazaraa, H.D. Sherali, and C.M. Shetty. *Nonlinear programming: Theory and Algorithms, third edition*. Wiley-interscience, Hoboken, New Jersey, 2006.

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

M.A. Boschetti, A. Mingozzi, and S. Ricciardelli. A dual ascent procedure for the set partitioning problem. *Discrete Optimization*, 5:735–747, 2008.

A. Charnes and W.W. Cooper. Programming with linear fractional functions. *Naval research logistics quart*, 9:181–186, 1962.

J.-F. Cordeau. A branch-and-cut algorithm for the dial-a-ride problem. *Operations Research*, 54(3):573–586, 2006.

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, 2003a.

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

J.-F. Cordeau and G. Laporte. The dial-a-ride problem: models and algorithms. *Annals of Operations Research*, 153(1):29–46, 2007.

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

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

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

W. Dinkelbach. On nonlinear fractional programming. *Management Science*, 13(7):492–498, 1967.

Y. Dumas, J. Desrosiers, and F. Soumis. Large scale multi-vehicle dial-a-ride systems. Technical report, GERAD, École des Hautes Études Commerciales, Montréal, 1989.

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.

T. Garaix, D. Josselin, D. Feillet, C. Artigues, and E. Castex. Point-to-point on-demand transportation in a rural areas. A route optimisation method. *Cybergeo*, 2007. A Selection of the best Articles (SAGEO 2005). Article 396 online : http://www.cybergeo.eu/index11373.html.

T. Garaix, C. Artigues, D. Feillet, and D. Josselin. Vehicle routing problems with alternative paths: an application to on-demand transportation. *European Journal of Operational Research*, 204(1):62–75, 2010.

P.C. Gilmore and R.E. Gomory. A linear programming approach to the cutting stock problem – part II. *Operations Research*, 11(6):863–888, 1963.

B. Golden, S. Raghavan, and E. Wasil, editors. *The Vehicle Routing Problem: Latest Advances and New Challenges*, volume 43 of *OR/CS interfaces*. Springer, 2008.

B. Jaumard, P. Hansen, and M. Poggi de Aragao. Column generation methods for probabilistic logic. *ORSA Journal on Computing*, 3:135–148, 1991.

D. Josselin and C. Genre-Grandpierre. Les transports à la demande pour répondre aux nouvelles formes de mobilité. le concept de modulobus. In B. Montulet, M. Hubert, C. Jemelin, and S. Schmitz, editors, *Mobilités et temporalités*, pages 151–164. Facultés Universitaires Saint-Louis, Bruxelles, 2005.

L.S. Lasdon. *Optimization Theory for Large Systems*. Macmillan series in operations research, 1970.

H. Li and A. Lim. A metaheuristic for the pickup and delivery problem with time windows. In *13th IEEE international conference on tools with artificial intelligence (ICTAI) 2001, Dallas, USA*, pages 160–170, 2001.

S. N. Parragh, K. F. Doerner, and R. F. Hartl. A survey on pickup and delivery problems. part I: Transportation between customers and depot. *Journal für Betriebswirtschaft*, 58(1):21–51, 2008a.

S. N. Parragh, K. F. Doerner, and R. F. Hartl. A survey on pickup and delivery problems. part II: Transportation between pickup and delivery locations. *Journal für Betriebswirtschaft*, 58(2):81–117, 2008b.

S.N. Parragh, J.-F. Cordeau, K.F. Doerner, and R.F. Hartl. Models and algorithms for the heterogeneous previous termdial-a-ridenext term problem with driver related constraints. Technical report, University of Vienna, Faculty of Business, Economics and Statistics, 2009.

S. Ropke and J.-F. Cordeau. Branch-and-cut-and-price for the pickup and delivery problem with time windows. *Transportation Science*, 43(3):267–286, 2009.

S. Ropke, J.-F. Cordeau, and G. Laporte. Models and branch-and-cut algorithm for pick-up and delivery problems with time windows. *Networks*, 49(4):258–272, 2007.

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

M. Sol. *Column generation techniques for pickup and delivery problems*. PhD thesis, Technische Universitet Eindhoven, 1994.