European Journal of Operational Research 238 (2014) 300–312

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Innovative Applications of O.R.

Branch-and-price and constraint programming for solving a real-life technician dispatching problem

Cristián E. Cortés [a,*], Michel Gendreau [b], Louis Martin Rousseau [b], Sebastián Souyris [c], Andrés Weintraub [d]

[a] Civil Engineering Department, Universidad de Chile, Blanco Encalada 2002, 5th Floor, Santiago, Chile
[b] CIRRELT and MAGI, Ecole Polytechnique de Montréal, C.P. 6079, Succ. Centre-ville, Montreal, Quebec H3C 3A7, Canada
[c] McCombs School of Business, University of Texas at Austin, 2110 Speedway Stop B6500, Austin, TX 78712, USA
[d] Industrial Engineering Department, Universidad de Chile, Republica 701, Santiago, Chile

ARTICLE INFO

Article history:
Received 13 April 2013
Accepted 1 March 2014
Available online 13 March 2014

Keywords:
Branch-and-price
Constraint programming
Routing
Technician dispatch problem

ABSTRACT

We consider a real problem faced by a large company providing repair services of office machines in Santiago, Chile. In a typical day about twenty technicians visit seventy customers in a predefined service area in Santiago. We design optimal routes for technicians by considering travel times, soft time windows for technician arrival times at client locations, and fixed repair times. A branch-and-price algorithm was developed, using a constraint branching strategy proposed by Ryan and Foster along with constraint programming in the column generation phase. The column generation takes advantage of the fact that each technician can satisfy no more than five to six service requests per day. Different instances of the problem were solved to optimality in a reasonable computational time, and the results obtained compare favorably with the current practice.

1. Introduction

This research was motivated by a real problem, which is the dispatch of technicians from a large company that provides repair services for their machines to overcome failures that occur over a typical working day. The company offers black-and-white digital printers, digital presses, multifunction devices, and digital copiers. Of the services that the firm provides to its clients, the maintenance of its machines distributed over Santiago is probably one of the most important for building a positive image of the company in terms of level and quality of service over time. It is therefore critical to carefully route the available technicians to satisfy the client requirements occurring in certain service areas on a typical working day. In this research, we propose a formulation and solution for this problem, satisfying the typical routing constraints as well as soft time-window constraints. The latter constraints arise from the fact that service requests come from clients with different priorities. The company defines different promised service times (denoted hereinafter target service times) according to the importance of each customer. These promises of technician arrival can be violated, so these conditions can be added as penalties in the objective function. From this, we see that the proposed scheme is based on the classical formulation of the vehicle routing problem with soft time windows (VRPSTW), which is formulated and solved using constraint programming (CP)-based column generation (Section 3).

The vehicle routing problem (VRP) involves constructing routes for a set of vehicles to serve a set of customers, given a number of requirements. There are many classes of the VRP, depending on the requirements and constraints.

One of the best-known versions is the vehicle routing problem with time windows (VRPTW), which has the characteristic that each of the clients must be served within a predefined time interval. VRPTW has been one of the most intensively studied NP-hard problems in the last decades. Exact methods are still restricted to solve a limited number of real instances, and their performance strongly depend on the time-window characteristics. Heuristic and meta-heuristic approaches have been developed for many cases and potential applications of VRPTW. A complete review of the models and methods to solve the VRPTW can be found in Cordeau, Desaulniers, Desrosiers, Solomon, and Soumis (2002, chap. 7). Later on, Bräysy and Gendreau (2005a, 2005b) performed deep reviews of VRPTW solution methods, focused on the description of heuristic route constructions methods, local search algorithms and meta-heuristic methods for solving capacitated problems of different size and time-window configurations. Nowadays, state of the art results are currently offered by the hybrid

* Corresponding author. Tel.: +56 2 29784380; fax: +56 2 26894206.
E-mail addresses: ccortes@ing.uchile.cl (C.E. Cortés), michel.gendreau@cirrelt.ca (M. Gendreau), louis-martin.rousseau@polymtl.ca (L.M. Rousseau), sebastian.souyris@utexas.edu (S. Souyris), aweintra@dii.uchile.cl (A. Weintraub).

http://dx.doi.org/10.1016/j.ejor.2014.03.006
0377-2217/

genetic algorithm due to Nagata, Bräysy, and Dullaert (2010). Their method combines powerful route minimization procedures, proposing an effective edge assembly crossover along with very efficient local search algorithm. In the same line of research, Ibaraki et al. (2008) and Pisinger and Ropke (2007) developed two simple, efficient and flexible methods able to address various VRP variants, namely the Iterated Local Search (ILS) method and the Adaptive Large Neighbourhood Search (ALNS), respectively. Recently, Vidal, Crainic, Gendreau, and Prins (2013) proposed an efficient Hybrid Genetic Search with Advanced Diversity Control for a large class of time-constrained vehicle routing problems, adding to the method new features to properly handle the temporal dimension.

The most successful exact methods are based on column generation for linear programs with decomposable structures; this was originally introduced in Dantzig and Wolfe (1960). The first application of this approach to the VRP was developed by Desrochers, Desrosiers, and Solomon (1992) (see also Desrosiers, Solomon, & Soumis, 1995, chap. 2). The basic idea is to decompose the problem into sets of customers visited by the same vehicle (routes) and to select the optimal set of routes among all possible routes. The decomposition is based on two structures: a master problem and a subproblem. The former optimizes the global route objective function and includes the constraint that each customer must be covered exactly once, resulting in a set partitioning formulation. The latter creates new routes that improve the solution, using dynamic programming to solve a shortest path problem with time windows. More recently column generation methods have been extended to include cuts, heuristic pricing and relaxed elementarity constraints (Desaulniers, Lessard, & Hadjar, 2008) to produce better lower bounds for the master problem and to speed up the generation of columns. Exact solution approaches using column generation without requiring Branch and Price have been proposed by Baldacci, Mingozzi, and Roberti (2012).

With regard to vehicle routing schemes for technician dispatch problems, it is worth mentioning a real application of technicians dispatch for emergency calls of an important electricity company in Chile (Weintraub, Abud, Fernandez, Laporte, & Ramirez, 1999). Blakeley, Bozkaya, Cao, Hall, and Knolmajer (2003) and Weigel and Cao (1999) also develop and implement technician dispatch systems for large companies using heuristics and GIS data. Recent publications related to service technician routing and scheduling are not many. We can mention the work by Cordeau, Laporte, Pasin, and Ropke (2010), who address a service technician scheduling problem arising in large telecommunications companies, focusing on team configurations and assignment of tasks respecting specific skills, priorities and precedence constraints for the tasks assigned, with the objective of minimizing a weighted function related to the makespan. Xu and Chiu (2001) also perform task scheduling for a telecommunication company with the objective of maximizing the realized orders. Tang, Miller-Hooks, and Tomastik (2007) implement a tabu search heuristic for a real world maintenance dispatch system which is formulated as a Multiple Tour Maximum Collection Problem with Time-Dependent rewards. Liberatore, Righini, and Salani (2011) present an exact solution procedure for the VRPSTW. The proposed algorithm is a column generation scheme, with ad hoc heuristics based on dynamic programming to generate new columns. We use CP for that purpose instead. Liberatore et al. (2011) model explicitly the time window violation for early and late arrivals. By contrast, in our work we just penalize for late arrivals due to the nature of the real application we are trying to solve; nevertheless, in our model and solution algorithm it is straightforward to incorporate penalization for early arrivals as explained later in Section 3.2. Kovacs, Parragh, Doerner, and Hartl (2012) formulate a service technician routing and scheduling problem motivated by a real problem faced by infrastructure service and maintenance providers. The objective is to minimize the sum of total routing and outsourcing costs, considering hard time windows for reaching customers sites. They solve the problem through an Adaptive Large Neighborhood Search algorithm, tested on both artificial and real-world instances. Pillac, Gueret, and Medaglia (2013) adapt the large neighborhood search algorithm (Shaw, 1998) for the VRPTW with resource constraints. Finally, Souyris, Cortés, Ordoñez, and Weintraub (2013) formulate a robust optimization version of a column generation scheme for a technician dispatch problem, putting the focus on properly handling the uncertainty associated with service time in such kind of real applications.

The CP-based column generation (CG) framework was introduced by Junker, Karisch, Kohl, Vaaben, Fahle, et al. (1999) and Yunes, Moura, and de Souza (2000) for two different crew rostering problems. Their work was motivated by the difficulties that arose in standard column generation approaches when modeling complex rules from legislation and union agreements. To overcome these difficulties, they proposed solving the pricing subproblem using CP models of resource-constrained shortest paths on acyclic graphs. The most attractive feature of CP, compared with dynamic programming, was the expressiveness of its modeling languages. Rousseau, Gendreau, Pesant, and Focacci (2004) and then Chabrier (2006) extended this framework to cyclic graphs, thus allowing VRPTW problems to be solved. In the last decade, the CP-CG framework has been used in several applications that are discussed in a recent survey by Gualandi and Malucelli (2009).

In this paper we use a CP-based pricing model similar to that proposed by Yunes et al. (2000) and Yunes, Moura, and De Souza (2005) in the context of bus driver scheduling, which allows for a flexible and simple modeling of the technician-route requirements. This straightforward model, which takes advantage of the fact that each technician visits only a small fraction of the overall daily clients, has significantly fewer variables than the graph-inspired models normally used for routing problems (Rousseau et al., 2004). Although it may be somewhat less efficient, this model does not require the implementation of complex shortest-path constraints such as those proposed for airline crew rostering problems (Junker, Karisch, Kohl, Vaaben, Fahle, et al., 1999). To obtain optimal solutions for different service request distributions, we implement a branch-and-price approach based on the efficient branching strategy of Ryan and Foster (1981), again taking advantage of CP flexibility when we include these branching constraints.

The main contributions of this paper are threefold: (i) we provide an efficient exact solution procedure for the VRPSTW using CP; (ii) it is the first to show that the approach proposed by Yunes et al. (2000, 2005) for crew scheduling problems can be applied to problems whose underlying structure is a cyclic graph, and to integrate it into a branch-and-price framework; (ii) as stated above, the approach that we propose has some advantages over the alternatives; it can thus prove attractive for similar practical applications.

The remainder of this paper is organized as follows. In the next section we state the problem formally and provide a mathematical formulation. Section 3 presents the column generation approach, while the branch-and-price methodology is described in Section 4. Experiments on a real-case scenario are reported in Section 5.

2. Problem statement and mathematical model

The strategic objective of the firm is based on client satisfaction. Within this context, the maintenance of their machines is one of the most important activities of the company in Chile. As explained in Section 1, service requests have different priorities, and there are different target response times for service requests at different

priority levels. Strictly, the target response time is defined as the maximum allowable time for a technician to reach the service location, measured from the time of the service request. If the technician reaches the location after the target response time, a penalty will be incurred by the system; this is considered in the formulation.

To consider this effect, we use a compound objective function for assigning technicians and jobs. The function minimizes two components: the sum of the differences between the target response times of requests and the effective service times provided by the firm, plus the travel times. This objective function seems to meet the needs of the company since it takes into account both service quality and the effective use of technicians.

The approach considers expected travel and service times, which are estimated from historical company data. In further developments, we will include uncertainty in the service times to make the model more robust and realistic. Travel times are less important than service times in the results, and therefore the assumption of deterministic travel behavior is reasonable in this case.

The set of service requests assigned during a given day come from the previous days, usually the day before, since the company attempts to enforce a 24-hour service policy. It is assumed that the dispatcher in charge of service selects which requests should be handled during the coming day. Furthermore, the dispatcher will choose a set of high-priority requests to be served first. A technician will begin his working day at one of these high-priority customer locations, under the reasonable assumption that the number of service requests is larger than the number of available technicians. In some cases, a technician can start the working day at the depot because he was not assigned to a specific client. In addition, all the tasks assigned to each technician must be completed during the day.

The modeling scheme was adapted from the classical formulation of the VRPSTW (Cordeau et al., 2002, chap. 7) with the following adjustments:

- For this problem, only the upper bound of the soft time window (related to the target response time of each service request) is considered, since the objective is to serve each requirement as soon as possible.
- Service times are quantitatively longer and are known less accurately than travel times; therefore, the former will play a more important role than the latter in the proposed decision rules.
- Although the VRPSTW considers soft time windows by penalizing the objective function, we add the hard constraint that all tasks scheduled for a specific day must be completed during that day. To ensure feasibility, we modified the classical formulation slightly to be able to decide which service requests can be handled during the day by the available technicians; any remaining requests are then postponed to the next day and given a very high priority.

Let $\mathcal{K} = \{1, \ldots, K\}$ be the set of technicians available to work during the day. As mentioned earlier, the technicians must start the day at the location of a service request with a very high priority or at the depot. Let $I_1 = \{1, \ldots, K\}$ be the set of these locations; and $I_2 = \{K+1, \ldots, C\}$ be the set of locations of the service requests that remain to be scheduled, where C is the cardinality of $I_1 \cup I_2$. We set $C+1$ as a dummy depot to which all technicians must be sent after their schedule is finished. To simplify the notation, we define the arc set $A = \{(i,j) | i \in I_1 \cup I_2, j \in I_2 \cup \{C+1\}, i \neq j\}$ to represent all feasible trips between locations. Additionally, let b_i be the time-window upper bound of service request i, and s_i the expected repair time for service request $i, i \in I_1 \cup I_2$. Let t_{ij} be the travel time from service request i to service request j, $(i,j) \in A$. We assume that travel times satisfy the triangular inequality. The end of the working day is set at instant F, which represents the latest time a technician can start the last job of the day. $\beta \in [0,1]$ is a multiobjective parameter.

We next formulate a mixed integer model, including both {0–1} (binary) and continuous variables. Specifically, we define flow variable x_{ij}^k, which is equal to 1 if technician k attends service request i and service request j sequentially, and 0 otherwise, $(i,j) \in A,\ k \in \mathcal{K}$. Two continuous variables are also included: w_{ik} is the time that technician k arrives at the location of service request $i, i \in I_1 \cup I_2 \cup \{C+1\},\ k \in \mathcal{K}$; and δ_{ik} represents the amount of time by which technician k violates the soft time window of service request $i, i \in I_2,\ k \in \mathcal{K}$. Finally, we define the binary variable v_i, which allows the model to handle the demand not served during the day with the available fleet by scheduling those service requests for the beginning of the next day, $i \in I_2$. In the model, such requests are assigned to virtual paths at a high penalty P. Thus, v_i is equal to 1 if i is sent through a virtual path, 0 otherwise. To simplify the notation we define the sets $O(i)$ and $D(i)$ associated with node i: $O(i) = \{j | (i,j) \in A\},\ i \in I_1 \cup I_2$, and $D(i) = \{j | (j,i) \in A\},\ i \in I_2 \cup \{C+1\}$. The formulation is as follows:

$$\min_{x,w,v,\delta} \quad \beta \sum_{k \in \mathcal{K}} \sum_{i \in I_2} \delta_{ik} + (1-\beta) \sum_{k \in \mathcal{K}} \sum_{(i,j) \in A} t_{ij} x_{ij}^k + \sum_{i \in I_2} P v_i \tag{1}$$

$$\text{s.t.} \quad \sum_{k \in \mathcal{K}} \sum_{j \in O(i)} x_{ij}^k = 1 \qquad i \in I_1 \tag{2}$$

$$\sum_{k \in \mathcal{K}} \sum_{j \in O(i)} x_{ij}^k = 1 - v_i \qquad i \in I_2 \tag{3}$$

$$\sum_{j \in O(i)} x_{ij}^k - \sum_{j \in D(i)} x_{ji}^k = \begin{cases} 1 & i = k \\ -1 & i = C+1 \\ 0 & \text{otherwise.} \end{cases} \qquad i \in I_1 \cup I_2 \cup \{C+1\},\ k \in \mathcal{K} \tag{4}$$

$$w_{ik} + s_i + t_{ij} - w_{jk} \leqslant \left(1 - x_{ij}^k\right) F \qquad (i,j) \in A,\ k \in \mathcal{K} \tag{5}$$

$$w_{ik} \leqslant F \sum_{j \in D(i)} x_{ji}^k \qquad i \in I_2,\ k \in \mathcal{K} \tag{6}$$

$$w_{ik} - \delta_{ik} \leqslant b_i \qquad i \in I_2,\ k \in \mathcal{K} \tag{7}$$

$$x_{ij}^k \in \{0,1\} \qquad (i,j) \in A,\ k \in \mathcal{K} \tag{8}$$

$$v_i \in \{0,1\} \qquad i \in I_2 \tag{9}$$

$$w_{ik}, \delta_{ik} \geqslant 0 \qquad i \in I_2,\ k \in \mathcal{K}. \tag{10}$$

The objective function (1) accounts for the total cost, computed as a convex combination of the sum of soft time-window violations and the total travel time. An additional term is added to penalize the unsatisfied demand. Constraints (2) and (3) restrict the assignment of each service request scheduled during the day to exactly one technician; if the request is postponed to the next day, the condition $v_i = 1$ together with constraint (3) ensure that no technician attends it on the current day. Next, constraints (4) ensure flow conservation, considering the dummy node where each path must end. Constraints (5) guarantee schedule feasibility with respect to precedence-time consistency (i.e., if technician k travels from customer i to customer j, then the arrival time at node j will be greater than or equal to the arrival time at node i plus the service time there). If $x_{ij}^k = 1$, then k goes from i to j, and we can start serving j no earlier than the start of service at i, plus the time spent in i, plus the travel time from i to j. If $x_{ij}^k = 0$, technician k does not go from i to j and constraint (5) becomes inactive. Note that, for a given k, constraints (6) force $w_{ik} = 0$ if service request i is not met by technician k and require k to reach i before time F if service is going to be provided (i.e., $v_i = 0$). Constraints (7) define δ_{ik}, the violation of the soft time window. Finally, (8)–(10) impose binary conditions on the flow variables and state the nonnegativity restriction on the time-arrival variables.

The VRP problem with soft time windows is hard to solve as mentioned in Taillard, Badeau, Gendreau, Guertin, and Potvin

(1997). We propose to solve this problem using column generation, as described in the next section.

3. Column generation approach

By working with actual company data, we realized that most real instances were not solvable using the arc-based formulation presented in the previous section, because the number of variables was too large to handle. Therefore, we looked for an alternative formulation and solution approach.

Column generation approaches have provided promising results for various types of VRPs (see for example Desrochers et al., 1992). Moreover, column generation approaches are very flexible in the sense that the problem under consideration can be split into two parts: a main model, the master problem, which chooses the routes with the minimum total cost from a pool of feasible routes; and a secondary model, the subproblem, which generates feasible routes that could potentially reduce the total cost.

In the literature, the subproblem is usually solved via dynamic programming (DP). We decided to use CP instead, for several reasons. First, it seems that CP works well when the length of each route is relatively small, which is the case for this application. Second, the CP model we implement is simple and easy to code. Third, under a CP approach, any additional constraint can be incorporated directly into the code without a special modeling technique. This is crucial when we use the branch-and-price algorithm (see Section 4) improved by the branching strategy proposed by Ryan and Foster (1981). Imposing some constraints on routes turned out to be straightforward with CP. For example, the proposed branching strategy forces two specific service requests to appear in a specific route, which can be included easily under CP, whereas in a DP implementation incorporating such constraints could become cumbersome. In Sections 3.2 and 4 we describe these features in detail.

3.1. Master problem

The master problem can be formulated as a set partitioning model assuming that it is possible to choose routes for each technician from an existing set of routes R. We use the result presented in Barnhart, Johnson, Nemhauser, Savelsbergh, and Vance (1998), which shows that the set covering relaxation is numerically far more stable and thus easier to solve than the set partitioning version. Also the authors argue that it is easy to construct a feasible integer solution from a solution of the set covering relaxation.

Each route $r \in R$ is characterized by a technician who initiates the route r at a specific service location $i_1 \in I_1$, and then follows a sequence of service requests $\{i_2, \ldots, i_e\} \subseteq I_2$, where i_2 is the second one and i_e is the service request at the end of route r. For each service request in position $l \in \{2, \ldots, e\}$ of the route, the technician's arrival time is $w_{i_l} = w_{i_{l-1}} + s_{i_{l-1}} + t_{i_{l-1}i_l}$, and thus the time-window violation becomes $d_{i_l} = \max\{0, w_{i_l} - b_{i_l}\}$. Additionally, $w_{i_1} = d_{i_1} = 0$. Hence, the total cost of the route is $c_r = \beta\sum_{l=2}^{e} d_{i_l} + (1-\beta)\sum_{l=1}^{e-1} t_{i_l i_{l+1}}$. In this formulation, the binary variable θ_r indicates whether or not route $r \in R$ is chosen. a_{ir} is a binary parameter that indicates whether or not route r contains service request i, and the binary variable v_i is equal to 1 if service request i is not included in any route, incurring a high penalty P. Then the master problem is the following:

$$(\text{MP}) \quad \min_{\theta, v} \quad \sum_{r \in R} c_r \theta_r + \sum_{i \in I_1 \cup I_2} P v_i \tag{11}$$

$$\text{s.t.} \quad \sum_{r \in R} a_{ir}\theta_r + v_i \geqslant 1 \quad i \in I_1 \cup I_2 \tag{12}$$

$$\theta_r \in \{0,1\} \quad r \in R \tag{13}$$

$$v_i \in \{0,1\} \quad i \in I_1 \cup I_2. \tag{14}$$

Note that in the MP formulation, the pool R of routes can be empty, and there still exists a feasible solution: $v_i = 1,\ i \in I_1 \cup I_2$, with cost $C \times P$, where C is the cardinality of $I_1 \cup I_2$ as defined before.

To generate new columns, we replace the integrality constraints (13) and (14) by $\theta_r \in [0,1],\ r \in R$, and $v_i \in [0,1],\ i \in I_1 \cup I_2$, respectively. Thus, for a given pool of routes R, the optimal solution of MP provides the dual values of constraint (12) to the subproblem that generates new routes. This procedure is described below.

3.2. Subproblem

Given a pool R of columns and the associated optimal solution of the LP relaxation of problem (11)–(14), it is well known from linear programming theory that a new column r, not in R, has the potential to improve the objective function only if it has a negative reduced cost. The reduced cost of a column is defined as the cost of the route, c_r, minus the sum of the dual variables of the service requests that belong to that route, which come from constraint (12).

At each iteration, the subproblem identifies a route of minimum reduced cost. We solve the master problem over the current set of columns using the simplex method and obtain the dual variables. Let α_i be the dual variables associated with constraint (12). The subproblem generates the optimal route by minimizing the real cost c_r (the sum of the time-window violations and travel times) minus the sum of the dual variables of the service requests included in the route, subject to the constraints that ensure that the route is feasible. By this process, a route with minimum reduced cost is generated. If the resulting reduced cost is negative, this new column can be included in the basis to improve the objective function of the master problem.

Constraint programming models have been used to solve pricing subproblems within hybrid column generation for almost a decade (Gualandi & Malucelli, 2009). There are essentially three families of models in the literature that specifically address routing and scheduling problems. One of the first models, proposed by Yunes et al. (2000, 2005), is based on an array of finite-domain variables $X_p \in T,\ p \in P$, that identify which task in T is to be performed by a bus driver in position $p \in P$. Junker, Karisch, Kohl, and Vaaben (1999); Junker, Karisch, Kohl, Vaaben, Fahle, et al. (1999) and Fahle et al. (2002) proposed using a single set variable $S \subseteq T$ to identify the subset of tasks in T to be covered by a crew pairing. In order for their model to be valid and efficient, they introduced a new shortest path constraint, as well as a negative-reduced-cost constraint on S. Finally, in the context of vehicle routing with time windows, Rousseau, Pesant, and Gendreau (2001), Rousseau et al. (2004) based their model on successor variables $N_t \in T,\ t \in T$, that identify the task to be performed immediately after t. This approach also required the use of specially designed global constraints and a search strategy based on dynamic programming.

In this paper, we adapt the simple model proposed by Yunes et al. (2000) for the following reasons. First of all, it is straightforward, is flexible, and works without the addition of dedicated global constraints. In constraint programming, a global constraint is a relationship among decision variables for which an efficient algorithm is available that can find the set of all infeasible values for each of the included decision variables. For example, the global constraint alldifferent(x_1, x_2, x_3) ensures that the three variables x_1, x_2, and x_3 are all different. Modern constraint programming packages offer a set of global constraints, and associated algorithms, that can model a generous number of relationships. The algorithms to find feasible sets are based on the principles of constraint propagation and domain reduction. When it is not possible to model a particular relationship with the existing global constraints, it is necessary to design a dedicated global constraint

and an associated algorithm. A detailed explanation of these concepts is found in the exhaustive survey by Régin (2011).

Second, in contrast with the model of Rousseau et al. (2004), which defines one variable for every node or task in the problem, Yunes et al. (2000) introduce only a number of variables equivalent to the number of tasks that can be performed by one technician. Since in our context this number is small (from three to six), this model seems particularly appropriate. Moreover in both these models the size of domains of the main decision variables are equal as they correspond to the number of nodes in the graph.

In the subsequent model (15)–(25), the new route to be generated is represented by the array of variables $sc[l],\ l = 1,\ldots,L$, where L is the maximum number of service requests that a technician can satisfy in one day. The lth element of sc is the service request scheduled in position l of the route. The route must start at an initial location of a technician (which could be either a high-priority service request or the depot, in case there are more available technicians than high-priority requests), that is a location from I_1. The route must contain at least one service request that remains to be scheduled, that is a service request from set I_2. Any position of the route that is not used by a service request must be utilized by fictitious nodes that are added for CP modeling purposes. Hence, the maximum number of fictitious nodes that a route can contain is $L-2$. Let us define the set of fictitious nodes to be $I_3 = \{C+1,\ldots,C+L-2\}$, where, as before, C is the cardinality of $I_1 \cup I_2$. Therefore, the domain of the $sc[l]$ variables, $l = 1,\ldots,L$, is set to $I_1 \cup I_2 \cup I_3$. The distinction between the fictitious nodes in I_3 is necessary in order to impose the global constraint alldifferent(sc), which will be discussed after the presentation of the model. The travel time between a service request in I_2 and any fictitious node in I_3 is set to 0, as is the travel time between any pair of nodes in I_3. Fig. 1 shows an example of a route that starts servicing request 3, then proceeds to requests 5, 10, and 14, and finishes with the fictitious nodes $C+1$ and $C+2$.

Variables $w[l]$ and $d[l]$ define the start time of service and the time-window violation respectively for the service request in position $l = 1,\ldots,L$. Variable $a[i],\ i \in I_1 \cup I_2 \cup I_3$ takes the value 1 if request i is served by the route, and 0 otherwise, $l = 1,\ldots,L$. These definitions are useful for the control of the branch-and-price method as described in Section 4. The constraint programming model for the subproblem is thus the following:

$$(\text{SP}) \quad \min \quad \beta\sum_{l=1}^{L} d[l] + (1-\beta)\sum_{l=1}^{L} t_{sc[l-1],sc[l]} - \sum_{l=1}^{L} \alpha_{sc[l]} \tag{15}$$

$$\text{s.t.} \quad w[1] = 0 \tag{16}$$

$$w[l] = w[l-1] + s_{sc[l-1]} + t_{sc[l-1],sc[l]} \quad l = 2,\ldots,L \tag{17}$$

$$d[l] = \max(0, w[l] - b_{sc[l]}) \quad l = 1,\ldots,L \tag{18}$$

$$\text{alldifferent}(sc) \tag{19}$$

$$sc[l] \in I_1 \quad l = 1 \tag{20}$$

$$sc[l] \in I_2 \quad l = 2 \tag{21}$$

$$sc[l] \in I_2 \cup I_3 \quad l = 3,\ldots,L \tag{22}$$

$$sc[l] = i, i \in I_3 \Rightarrow sc[l+1] = i+1 \quad l = 3,\ldots,L \tag{23}$$

$$a[sc[l]] = 1 \quad l = 1,\ldots,L \tag{24}$$

$$\sum_{i \in I_1 \cup I_2 \cup I_3} a[i] = L \tag{25}$$

The objective function (15) minimizes the convex combination of time-window violation and travel cost, which is the real cost of the route minus the sum of the dual values, α, associated with the service requests in the route. For $i \in I_3$, which is not in constraint (12) of MP, we set α_i equal to 0. Constraint (16) ensures that the start time of the path is associated with the start time of the day, and (17) sets the starting time of next service requests that are visited by the route. CP constraint (18) sets the late time-window

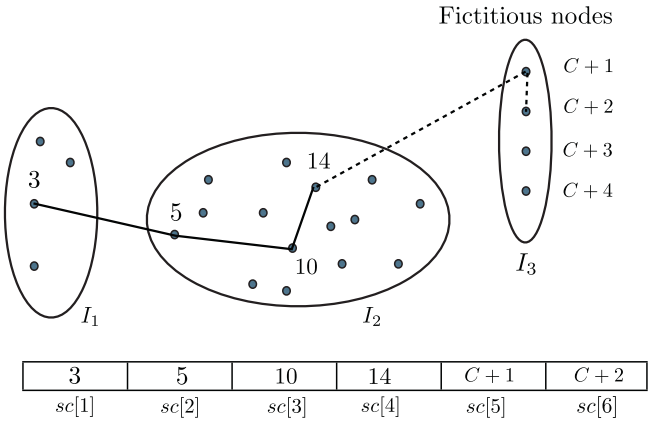


Fig. 1. Shortest path model implemented with CP.

violations. Note that the model can also include early time window violations; for adding that feature, we just have to change the right hand side of constraint (18) by $\max(f_{sc[l]} - w[l], w[l] - b_{sc[l]})$, where f_i corresponds to the time window lower bound associated with service request i. Thus, the CP algorithm presented next can be easily adapted to incorporate this modification. Constraint (19), alldifferent(sc), is a global constraint that ensures that all of the variables sc are different, so any request is served by the path no more than once (for an explanation of this global constraint see, for example, Régin, 2011). Constraint (20) ensures that the first request served by the route is assigned to a location where a technician is initially positioned, (21) ensures that the second position of the route is a service request without a technician initially allocated, and (22) restricts the other positions along the path to be either service requests or fictitious nodes. (23) ensures that, if position l is used by a fictitious node, the next service request in the route must be the next fictitious node (proceeding to any other service request is not allowed in order to reduce the search space as much as possible). Variable a is useful to impose new constraints for the service requests that can belong to a specific route, which are needed to implement the search procedure in the branch-and-price methodology, as explained below and detailed in Section 4. Finally, constraints (24) and (25) impose logical restrictions between the different variables in sc and a.

CG frameworks depend heavily on marginal costs to guide the search at the subproblem level. In some cases it is possible that, during the first iterations, the marginal cost associated with each customer is not accurately estimated by the dual values. For instance, in some routes some service requests pick up most of the total dual values. This undesirable behavior is illustrated by the example in Fig. 2. A path that visits each of overweighted service requests (2 and 3's dual values are exactly twice 1 and 4's dual values) will be considered a good route (with a reduced cost of −5), but it is not (it is unlikely to be selected in an optimal IP solution). With a more realistic distribution of the dual values, all

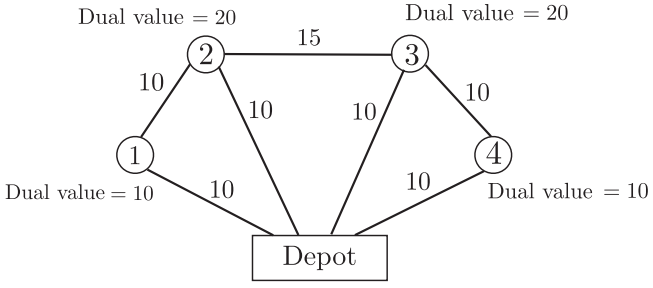


Fig. 2. Path *Depot* − 2 − 3 − *Depot* has a negative reduced cost of −5.

the nodes would have been given a value of 15 and no more reduced-cost paths would have been found, thus removing the need for a last iteration.

In this example, described in Rousseau, Gendreau, and Feillet (2007), such undesirable behavior may occur because the master problem is degenerate, and thus its dual has an infinite number of optimal solutions. The bases of the primal solutions all exhibit a common set of strictly positive variables, but different sets of null variables. A different dual solution is associated with each of these equivalent primal optimal solutions.

The standard function that returns dual values in the LP codes returns an extreme point of the dual polyhedron. Extreme solutions are characterized by large values for some marginal costs while others are at zero. Therefore, the subproblem tends to build routes that have very low reduced costs but potentially large travel times and time-window violations. To avoid this, we instead optimize over the real objective function (c_r). Negativity of the reduced cost can then be enforced through a constraint, since the necessary condition imposed by the CG framework to ensure convergence is that we add, at each iteration, a set of negative-reduced-cost routes or prove that none exists. The objective function (15) thus becomes (26) and constraint (27) is added to the subproblem model:

$$\min \beta\sum_{l=1}^{L} d[l] + (1-\beta)\sum_{l=1}^{L} t_{sc[l-1],sc[l]} \tag{26}$$

$$\beta\sum_{l=1}^{L} d[l] + (1-\beta)\sum_{l=1}^{L} t_{sc[l-1],sc[l]} - \sum_{l=1}^{L}\alpha_{sc[l]} < 0 \tag{27}$$

After experimenting with our CP implementation, we decided to split the subproblem into a two-step procedure (SP1 and SP2); we found a considerable reduction in the running time when many interesting columns are generated at each iteration of the pricing problem. SP1 attempts to select service requests such that the reduced cost is negative (i.e., constraint (27) is satisfied), but the order of the service requests in the route is not necessarily the best. Thus, SP1 is not an optimization problem but a CP feasibility problem with feasible region (17)–(27). SP2 then optimizes the sequence to find the lowest real cost given by (26) subject to (17)–(25). SP1 passes the selected service request to SP2 by fixing the variables a in constraint (24). Fig. 3 shows an example of the process. First, SP1 chooses service requests 0, 2, 3, 5, 8, and 10, and then SP2 rebuilds the route with only these service requests. Notice that the reduced cost of the resequenced route can only be lower than the original route or equal in case the sequence does not change, because the real cost of the re-sequenced route decreases and the sum of the dual variables remains the same.

Empirically, we found a considerable improvement in terms of the number of columns generated using this procedure, which reduces the computational time by about 30%.

To help SP1 find routes of good quality faster, we implement a simple but efficient search consistent with our problem. It guides the local search logically, by exploring a neighborhood matrix for each service request.

For a given set of dual values α, we define the neighborhood matrix $N(\alpha)$ to be a square matrix of C rows and columns (recall that $C = |I_1 \cup I_2|$). The element in row i and column p, n_{ip}, is the pth closest service request to request i. We considered different distance measures, and the one that performs the fastest in computational terms is the following. For a new route that has been created and includes the service request i, we would like to have the sequence (i,j) if two conditions hold: first, j's time-window upper bound, b_j, is close to $b_i + s_i + t_{ij}$ (recall that s_i is the expected service time of i, and t_{ij} is the expected travel time between i and j); and second, j has a large dual value. If j has a small dual value, then it is already efficiently covered by a route belonging to pool R. Thus, for each pair (i,j) we define the distance measure $d(i,j) = \gamma((b_j - (b_i + s_i + t_{ij}))^+ + \rho(b_i + s_i + t_{ij} - b_j)^+) + (1-\gamma)\frac{1}{\alpha_j}$, where $(x)^+ = \max\{0,x\}$, $\rho > 1$ and $\gamma \in [0,1]$. The larger penalization when $b_i + s_i + t_{ij} > b_j$ is to avoid a time-window violation at j. The implemented search for SP1 that uses $N(\alpha)$ is described in Algorithm 1.

Algorithm 1. SP1 constructs a new route with negative reduced cost, but not necessarily in the optimal order

Input : Dual values α from the master problem. Neighborhood matrix $N(\alpha)$
Output: A new route with negative reduced cost but not necessarily in optimal order

$sc[1] \leftarrow \arg\max_{i \in I_1}\{\alpha_i\}$;
$a[sc[1]] \leftarrow 1$;
for $l \leftarrow 2$ *to* L **do**
  $i \leftarrow sc[l-1]$;
  $p \leftarrow 1$;
  while $sc[l]$ *has no service request assigned to it* **do**
    if *it is feasible by* (17) *to* (27) *to assign service request* n_{ip} *to* $sc[l]$ **then**
      $sc[l] \leftarrow n_{ip}$;
      $a[sc[l]] \leftarrow 1$;
    else
      $p \leftarrow p + 1$;

We also notice an additional improvement in the performance of the subproblem when adding some redundant constraints that do not modify the solution set but improve the constraint propagation. These constraints improve the tree pruning, filtering processes, and domain reduction. We impose

$$sc[l] > first(I_3) \Rightarrow sc[l-1] = sc[l] - 1 \qquad l = 3,\ldots,L \tag{28}$$

$$sc[l] \leqslant first(I_3) \Rightarrow sc[l-1] < first(I_3) \qquad l = 3,\ldots,L \tag{29}$$

where $first(I_3)$ represents the first fictitious service request; i.e., $C+1$. Thus, constraints (28) and (29) eliminate identical solutions from the domain, taking into account the fact that all the fictitious nodes are the same, although their identification codes are different. Without these constraints, the solver does not realize early in the search tree that a service request cannot follow a fictitious node. In the CP literature, redundant constraints have often been used to create a set of different links between variables and to generate new potential for propagation.

In the implementation, instead of adding only one route per iteration of the column generation, we generate a number of routes with negative reduced costs. To do this, we take advantage of the CP search tree of SP1. Each final node of the search tree is a route solution that has a negative reduced cost; therefore, each has the potential to improve the solution of MP if it is added to the pool R. We add the first Q columns found. Between iterations, we tune Q depending on the time that SP1 is taking to find solutions. For the first iterations we set Q to a large number, and for the last iterations we set Q to 1.

We implemented a simple strategy to control the size of the pool of columns. At every iteration of the CG process, columns with high positive reduced costs are eliminated from the pool.

Finally, we implemented a branch-and-price method that guarantees finding the optimal solution. The method allows us to explore additional routes with negative reduced costs in the branch-and-bound tree, by inspecting each node of the tree rather than only the root node. The method performs significantly better when an ad hoc branching strategy is implemented, as explained below.

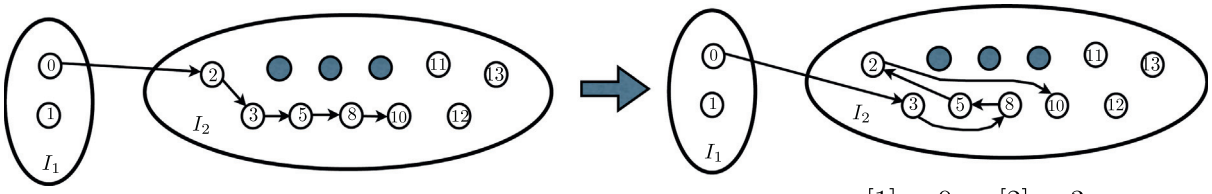
4. Branch-and-price implementation

Many successful implementations of branch-and-price schemes can be found in the literature. Barnhart, Hane, and Vance (2000) presented a CG model that is solved with a branch-and-price-and-cut algorithm for an origin–destination integer multicommodity flow problem. Savelsbergh and Sol (1991) solved a dynamic

$$sc[1] = 0;\ sc[2] = 2;\ sc[3] = 3;\ sc[4] = 5;\ sc[5] = 8;\ sc[6] = 10;$$

$$\Rightarrow \quad a[0] = a[2] = a[3] = a[5] = a[8] = a[10] = 1;\ a[4] = a[7] = a[9] = a[11] = a[12] = a[13] = 0;$$

Reconstruct the path with the nodes



$$sc[1] = 0;\ sc[2] = 3;\ sc[3] = 8;\ sc[4] = 5;\ sc[5] = 2;\ sc[6] = 10;$$

Fig. 3. Construction of a new route procedure.

vehicle routing problem with heuristic optimization techniques based on dynamic programming, together with a sophisticated column management scheme. Theoretical aspects of several versions of the branch-and-price algorithm are discussed by Barnhart et al. (1998), Vanderbeck (2000, 2005, 2006), and Vanderbeck and Savelsbergh (2006). Specifically, the authors concentrate on topics such as the formulation of the decomposition, proper ways to perform the branching, and column-management efficiency.

In this application, we use the branching strategy proposed by Ryan and Foster (1981) for generic set partitioning problems. Barnhart et al. (1998) proved the following proposition:

Proposition: If A is a 0–1 matrix, and a basic solution of $A\theta = 1$ is fractional i.e., at least one of the components of θ is fractional, then there exist two rows i and j in the master problem such that:

$$0 < \sum_{r:a_{ir}=1,a_{jr}=1} \theta_r < 1 \tag{30}$$

The pair $i,\ j$ establishes the following pair of branching constraints: $\sum_{r:a_{ir}=1,a_{jr}=1}\theta_r = 1$ and $\sum_{r:a_{ir}=1,a_{jr}=1}\theta_r = 0$, i.e., rows i and j must be covered by the same column on the first (left) branch and by different columns on the second (right) branch. The CP model can incorporate constraints of this type in an efficient way.

Thus, for the left branch, constraint (31) must be added to SP1 to restrict the columns to those containing either both service requests i and j, or neither of them:

$$a[i] + a[j] \neq 1 \tag{31}$$

Similarly, for the right branch, constraint (32) can be added to restrict the columns to those containing at most one of the service requests, either i or j:

$$a[i] + a[j] \leqslant 1 \tag{32}$$

Many possible pairs of service requests (i,j) could be chosen. For efficiency, we use a rule similar to that proposed by Vance et al. (1997) to choose a pair of service requests with a large probability of being covered by the same route in a good feasible solution of the IP. We compute for each pair (i,j) the scalar $f(i,j) = \sum_{r:a_{ir}=1,a_{jr}=1}\theta_r$. We choose the pair with the largest $f(i,j)$ for branching, and the depth-first search strategy is applied along the left branch, ensuring that the chosen pair of service requests will be on the same route. As mentioned before, the structure of the proposed CP model allows us to implement the branching strategy in a straightforward way.

Our branch-and-price procedure is terminated when either we find an acceptable gap or the running time exceeds three hours. The gap is computed as the ratio of the integral objective value minus the linear relaxation objective value, and the linear relaxation objective value ((upper bound − lower bound)/lower bound).

In the next section, we report some empirical results for different service-request sets, using real data for a typical day.

5. Implementation and empirical results

The model was coded in Ilog Concert Technology and solved using CPLEX 9.0 (Ilog, 2003a) for the master problem and SOLVER 6.0 (Ilog, 2003b) for the subproblems. We test our algorithms in an Intel Pentium M 1.5 gigahertz, with 2 gigabyte of RAM. In this section, the branch-and bound-method (B&B), the optimal branch-and-price scheme (B&P), and the linear relaxation at the root node are run for several real instances of different sizes and service-request configurations, all from real data provided by the firm.

The consistency of the CG approach was empirically checked by observing the convergence pattern toward the optimal solution in small problems. In fact, for small instances the optimal solution was obtained directly by solving the IP corresponding to the original arc-based formulation presented in Eqs. (1)–(10).

In our results, each technician was assigned to no more than six service requests per day. For a larger number of requests, it was necessary to reschedule some for the next day, since serving all the requests on the same day was not feasible.

5.1. Description of the experiments with real data

We used a data set for the southern region of Santiago, Chile. Let us denote this region AB, since it can be further split into two subregions, namely A and B. A normal day was considered in terms of the number of service requests and the number of technicians working. Thus, 41 calls were received in zone A with 10 technicians available, while 35 were received in zone B with 9 technicians available. Note that the company presently schedules service for the two subregions separately. In our tests, we consider three basic instances, one for region A, another for region B, and one for the entire region AB. Fig. 4 shows a map of the city and the two zones.

The travel times between service-request locations were estimated by dividing the complete region AB into microzones. We decided to use the administrative divisions of Santiago (the microzones then match what are denoted *comunas* by the Chilean authorities). This allows us to use historical information for average travel times between each pair of comunas and within each comuna. Microzones (comunas) are required because the travel times between service requests are not available. The expected service time is computed as the average service time reported by

Fig. 4. Santiago city with zones A and B.

Table 1
Objective functions of company operation and B&P algorithm for different β values.

Subregion	β	Objective function	
		Manual	B&P
A	0.3	813	613
	0.6	854	578
	0.9	896	491
B	0.3	1083	920
	0.6	1214	761
	0.9	1346	604

Table 2
Performance of the company operation compared with the B&P algorithm in terms of total time-window violation and total travel time for different β values.

Subregion	β	Manual		B&P	
		Δ	T	Δ	T
A	0.3	910	772	717	568
	0.6	910	772	486	717
	0.9	910	772	427	1065
B	0.3	1390	952	1325	746
	0.6	1390	952	594	1012
	0.9	1390	952	544	1147

the technicians for each type of request over the year. Finally, target response times (as defined in the first paragraph of Section 2) were also provided by the company, depending on the priority assigned to each service request. In most cases, the observed travel times followed the triangular inequality. If they did not, we perturbed them with the smallest possible amount necessary to satisfy the triangular inequality. As mentioned earlier, travel times are much smaller than service times, therefore this perturbation does not significantly change the solution.

We considered $\beta = 0.3,\ 0.6,\ 0.9$, which should cover the different priority levels a modeler would assign to the two components of the objective function (travel times versus time-window violations) according to the company's goals. In this scheme, if it is infeasible to insert a specific service request into any route, that request will be assigned for service at the beginning of the next day, with a cost-function penalty of 500. Thus, the first columns generated for each service request are conceptually similar to the virtual paths defined in the original model through the variables v_i to handle infeasibility. They are assigned a high penalty to provide an initial feasible solution to the master problem. Under these conditions, with the original IP model, Eqs. (1)–(10), we obtain the same optimal solution as that obtained by B&P. Unfortunately, this result can be verified only for very small instances, since the original arc-based formulation becomes intractable as the number of requests increases.

Under these conditions, we run our model using the B&P algorithm separately for each subregion (A and B), in order to compare our results with the manual dispatch used by the company (company operation) for each subregion. Table 1 compares the objective values of the results. Table 2 compares the results in terms of time-window violation and travel time for the entire system. In Table 2, column Δ reports the total time-window violation observed for all service requests on all prescribed routes, and column T reports the sum of the travel time over all vehicle routes.

Table 1 shows that the model improved performance for all instances (with improvements between 15% and 45%). We notice a more significant improvement for region A, probably because access and routing are more complicated in this region. Moreover, as β increases, the improvement provided by the model becomes more significant. This is not surprising since larger values of β place more emphasis on time-window violations in the objective function. Increasing β thus yields routes in which most technician arrival times fall in a range quite close to the target response times.

Table 2 shows that in both cases (zones A and B), the results are sensitive to the value of β. This is reflected in the ratio of time-window violations to travel times in each case (namely $SH = \frac{\Delta}{T}$). Thus, in the case of zone A (B) SH goes from 1.8 to 0.5 (from 1.9 to 0.4) when β increases from 0.3 to 0.9. From the results obtained for this particular instance, we can observe how important the selection of the parameter β could become in the final performance of the model, both in terms of time-window violations and travel times.

We also carried out a sensitivity analysis with respect to two other parameters, to test not only the quality of the solutions but also the performance of the B&P algorithm. The parameters are the number of technicians (for the A instance ranging from 6 to 11 and for the B instance from 5 to 10) and the upper bound for the time window assigned to each service request with respect to the actual limit (denoted $tw1$). The idea is to run the model under tighter time-window conditions. We do this by decreasing the upper bound by 15% to 40%, thus generating new upper levels, $tw2 = 0.85 \cdot tw1$ and $tw3 = 0.6 \cdot tw1$. Table 3 gives the results of these tests for instance A with $tw1$, $tw2$, and $tw3$, while Table 4 gives the results for instance B with $tw1$, $tw2$, and $tw3$.

When we integrate regions A and B, the problem has 76 service requests (instance AB). This allows testing on a larger instance and also allows the dispatcher to consider more flexible options than those provided by a zoning fixed a priori. Eventually, the zoning could be defined by a more formal method than the traditional subdivision of the service area by the company. Zoning may no longer be necessary (the company could completely rely on the optimization), or a clustering-type method could be used to define zones in a more systematic way. Table 5 gives the results of these experiments for 11, 13, 15, 17, 19, and 21 technicians, considering also the sensitivity with respect to the time-window upper bounds $tw1$, $tw2$, and $tw3$.

The columns of Tables 3–5 represent the following: (tw) the upper bound considered for time windows, (β) the objective weighting parameter, and (#Tec) the number of technicians. For each instance generated by a combination of these parameters, we report

Table 3
Results for instance A.

tw	β	#Tec	Objective			Gap [%]		# Columns		# Nodes		Time [seconds]	
			LR	BB	BP	BB-LR	BP-LR	BB	BP	BB	BP	MP	SP
		6	467	468	–	0.2	–	955	–	10	–	130	6327
		7	465	468	–	0.7	–	1771	–	79360	–	100	5203
	0.3	8	463	468	–	1.0	–	1767	–	1939	–	15	7533
		9	463	466	–	0.5	–	1908	–	34	–	30	9020
		10	463	466	–	0.5	–	1932	–	1599	–	22	9227
		11	463	466	–	0.5	–	1894	–	8594	–	633	8056
		6	267	276	–	2.8	–	995	–	179	–	11	4403
		7	266	277	276	3.0	3.0	1558	33	8587	3	45	4535
*tw*1	0.6	8	265	274	267	3.3	0.7	1464	73	36926	105	45	5510
		9	265	274	265	3.3	0.2	1670	89	39410	81	53	6364
		10	265	274	265	3.3	0.2	1685	71	37027	69	49	6057
		11	265	278	–	2.2	–	1683	–	14706	–	87	6173
		6	67	67	–	0.7	–	948	–	28	–	10	3480
		7	66	67	–	1.2	–	1292	–	3566	–	15	2488
	0.9	8	66	67	–	1.4	–	1230	–	3477	–	15	2739
		9	66	67	–	1.4	–	1331	–	4761	–	15	2457
		10	66	67	–	1.4	–	1334	–	7338	–	24	2237
		11	66	67	–	1.4	–	1334	–	7338	–	17	2675
		6	481	484	–	0.6	–	1134	–	83	–	9	3563
		7	479	481	–	2.2	–	1784	–	20	–	13	3237
	0.3	8	476	481	–	2.3	–	1564	–	341	–	13	3295
		9	470	481	–	2.4	–	1846	–	760	–	14	3200
		10	470	481	–	2.4	–	1842	–	3881	–	17	3342
		11	467	467	–	0.0	–	1906	–	0	–	111	5203
		6	279	292	282	4.8	1.0	1235	182	1444	89	13	3197
		7	269	276	–	2.4	–	1566	–	46	–	12	1609
*tw*2	0.6	8	269	275	–	2.3	–	1382	–	205	–	11	1378
		9	269	275	–	2.3	–	1645	–	643	–	13	2246
		10	268	275	–	2.4	–	1577	–	660	–	13	2483
		11	268	275	–	2.7	–	1637	–	4624	–	18	2894
		6	71	80	71	12.4	0.4	1362	83	869	111	12	2246
		7	67	69	–	2.4	–	1406	–	33	–	11	800
	0.9	8	67	69	–	2.1	–	1199	–	57	–	10	804
		9	67	69	–	2.2	–	1373	–	529	–	10	839
		10	67	69	–	2.2	–	1345	–	485	–	10	921
		11	67	69	–	3.0	–	1377	–	4737	–	14	1647
		6	2028	2074	–	2.5	–	1097	–	7	–	5	900
		7	687	730	730	7.1	7.1	2137	1612	3753	203	36	4649
	0.3	8	589	590	–	0.3	–	1675	0	1	–	6	702
		9	566	568	–	0.2	–	1550	0	1	–	6	561
		10	562	569	–	0.3	–	1348	0	64	–	5	422
		11	539	548	–	1.0	–	1402	0	120	–	6	556
		6	1790	1981	1981	10.7	10.7	1524	1616	263	119	6	4226
		7	647	710	697	9.9	7.8	2389	1398	392	163	11	4341
*tw*3	0.6	8	457	453	–	0.3	–	1951	–	0	–	8	747
		9	387	387	–	0.0	–	1708	–	0	–	7	405
		10	369	370	–	0.4	–	1427	–	49	–	7	329
		11	333	333	–	0.0	–	1270	–	0	–	5	261
		6	1697	1888	1888	11.2	11.2	1593	1659	823	57	72	3561
		7	604	609	–	0.8	–	2511	–	4	–	10	795
	0.9	8	313	316	–	0.7	–	2119	–	12	–	9	574
		9	204	206	–	0.7	–	1779	–	4	–	8	353
		10	167	167	–	0.1	–	1446	–	4	–	6	166
		11	119	117	–	0.3	–	1386	–	4	–	6	142

the objective function obtained by the linear relaxation at the root node, the integer solution obtained using the default B&B implemented in CPLEX with only the columns found at the root node, and the integer solution obtained with the proposed B&P procedure (columns LR, BB, and BP respectively). We then report the gaps between (BB-LR) and (BP-LR), the number of columns generated at the root node (BB), the number of columns generated during the B&P (discounting the columns generated at the root node), the number of nodes explored by the CPLEX B&B, and the number of nodes explored by the B&P algorithm. Finally, the total running time to solve the master problem (MP) and the subproblem (SP) for both steps, namely B&B and B&P, are reported in seconds.

When the gap between the B&B solution and the linear relaxation at the root-node solution (column BB-LR) was lower than 3%, we did not run the B&P procedure. The tables report "–" for these cases.

To assign the technicians manually, the dispatchers were forced to split the region into zones A and B. Comparing the results for instances A and B in Tables 3 and 4 with those for instance AB in Table 5, we can see that this division leads to solutions of signifi-

Table 4
Results for instance B.

tw	β	#Tec	Objective			Gap [%]		# Columns		# Nodes		Time [seconds]	
			LR	BB	BP	BB-LR	BP-LR	BB	BP	BB	BP	MP	SP
		5	1928	1943	–	0.8	–	333	–	3	–	1.13	105.80
		6	1418	1419	–	0.1	–	554	–	7	–	1.32	93.16
	0.3	7	923	923	–	0.0	–	613	–	0	–	1.24	76.12
		8	920	920	–	0.0	–	501	–	0	–	1.16	81.24
		9	920	920	–	0.0	–	524	–	0	–	1.30	76.98
		10	919	920	–	0.1	–	466	–	0	–	1.16	73.88
		5	1312	1316	–	0.3	–	441	–	14	–	1.25	96.61
		6	1260	1263	–	0.2	–	582	–	0	–	1.39	74.27
*tw*1	0.6	7	764	764	–	0.0	–	603	–	0	–	1.50	85.34
		8	762	762	–	0.0	–	608	–	0	–	1.39	71.45
		9	762	762	–	0.0	–	534	–	0	–	1.20	79.74
		10	761	761	–	0.0	–	528	–	62	–	1.50	98.69
		5	1118	1118	–	0.0	–	458	–	0	–	1.42	87.70
		6	1099	1100	–	0.1	–	620	–	23	–	1.56	60.28
	0.9	7	604	604	–	0.0	–	650	–	0	–	1.67	68.21
		8	609	604	–	0.0	–	580	–	2	–	1.31	63.41
		9	604	604	–	0.0	–	614	–	0	–	1.39	72.00
		10	607	604	–	0.0	–	541	–	29	–	1.30	67.98
		5	3102	3457	3454	11.5	11.4	224	53	2	7	1.20	164.87
		6	1451	1452	–	0.1	–	533	–	4	–	1.33	78.83
	0.3	7	950	954	–	0.4	–	525	–	7	–	1.39	61.75
		8	942	945	–	0.4	–	544	–	10	–	1.22	57.75
		9	942	945	–	0.3	–	476	–	12	–	1.25	64.73
		10	940	945	–	0.3	–	478	–	9	–	1.12	53.50
		5	2328	2422	2422	4.0	4.0	347	72	2	5	0.68	162.12
		6	1283	1283	–	0.0	–	545	–	0	–	1.32	72.99
*tw*2	0.6	7	786	790	–	0.5	–	655	–	10	–	1.90	78.59
		8	783	785	–	0.3	–	567	–	2	–	1.50	53.08
		9	783	785	–	0.3	–	506	–	13	–	1.40	63.45
		10	783	785	–	0.3	–	493	–	4	–	1.36	55.01
		5	2201	2322	2321	5.5	5.5	432	164	12	7	0.73	166.63
		6	1111	1111	–	0.0	–	620	–	1	–	1.59	56.86
	0.9	7	620	621	–	0.1	–	648	–	9	–	1.75	58.23
		8	619	620	–	0.1	–	621	–	9	–	1.92	55.86
		9	619	620	–	0.1	–	532	–	19	–	1.51	48.30
		10	619	620	–	0.1	–	531	–	0	–	1.47	57.11
		5	12000	12000	–	0.0	–	34	–	0	–	0.03	0.22
		6	2496	2521	–	1.0	–	368	–	2	–	0.61	47.02
	0.3	7	1224	1530	1530	25.0	25.0	465	3018	201	549	2.53	302.30
		8	1039	1040	–	0.1	–	405	–	2	–	0.66	35.14
		9	1030	1032	–	0.1	–	379	–	0	–	0.50	24.53
		10	1016	1017	–	0.1	–	367	–	2	–	0.60	29.61
		5	9584	9584	–	0.0	–	36	–	0	–	0.03	0.38
		6	2498	2838	2520	13.6	0.9	431	741	2445	193	2.81	507.39
*tw*3	0.6	7	1214	1438	1436	18.5	18.3	567	1070	222	245	1.66	726.06
		8	905	917	–	0.7	-	523	–	2	–	0.79	40.66
		9	879	879	–	0.0	–	435	–	0	–	0.97	48.94
		10	857	860	–	0.3	–	408	–	3	–	0.64	25.37
		5	9560	9560	–	0.0	–	43	–	0	–	0.14	0.47
		6	2448	2736	2736	11.8	11.8	437	31	3283	23	4.01	46.57
	0.9	7	1147	1328	1328	15.8	15.8	582	2386	114	409	1.55	154.74
		8	758	769	–	1.5	–	544	–	15	–	0.77	34.28
		9	704	708	–	0.5	–	490	–	3	–	1.06	32.64
		10	670	671	–	0.1	–	449	–	1	–	0.68	17.68

cantly lower quality in all cases. If, for example, we consider instance ($tw1, \beta = 0.9$) with 6 technicians in zone A and 5 in zone B (thus, a total of 11 for AB), we see that the objective-function value obtained is 66 for case A and 1117 for case B, while that for case AB is 194. This deterioration in the objective value is due to the loss of flexibility when each zone is managed independently, since in zone B many customers need to be postponed to the next day. One conclusion to draw from these results is the considerable advantage of having an implementation of the B&P algorithm that can find an optimal solution for the larger 70-customer problem.

We allowed a maximum of 300 seconds to solve the subproblem at each iteration. We mark with a $*$ those instances where this limit was reached before an optimal solution was found. It is clear that in these cases the bound obtained is not valid and the final solution is not optimal. As noted earlier, after solving the MIP with the columns found at the root node (B&B), if the gap between the linear relaxation at the root node and B&B was below 3%, we did not run the B&P. When used (28 of the 162 cases), the B&P algorithm reduced the gap by 3.5% on average. And the gap was reduced for more than 0.1% in 17 of the 28 cases. Most of the computational time was spent on the subproblem phase. Instance A took much longer than instance B (mainly because there are more service requests in instance A), and instance AB took the longest. Except in five cases (all from instance B) when the B&P procedure

Table 5
Results for instance AB.

tw	β	#Tec	Objective			Gap [%]		# Columns		# Nodes		Time [seconds]	
			LR	BB	BP	BB-LR	BP-LR	BB	BP	BB	BP	MP	SP
		11	955	968	–	1.4	–	2790	–	104	–	39	3871
		13	936	941	–	0.6	–	2022	–	2	–	64	10459
	0.3	15	917	931	–	1.6	–	2208	–	284	–	16	10800
		17	911	929	–	2.0	–	2103	–	284	–	12	10800
		19	911	929	–	2.0	–	1928	–	316	–	11	9200
		21	911	929	–	2.0	–	1137	–	455	–	23	10800
		11	567	576	–	1.6	–	3354	–	1920	–	53	3246
		13	528	534	–	1.1	–	3125	–	725	–	17	10800
*tw*1	0.6	15	528	528	–	0.1	–	1814	–	3583	–	19	6334
		17	521	526	–	1.0	–	2072	–	586	–	47	9098
		19	513	515	–	0.4	–	2149	–	87	–	13	10206
		21	513	515	–	0.4	–	1731	–	2994	–	15	9001
		11	194	198	–	1.9	–	3969	–	2071	–	44	8048
		13	162	165	–	1.9	–	3394	–	4223	–	39	10800
	0.9	15	152	155	–	2.0	–	2414	–	4880	–	27	10365
		17	151	153	–	1.4	–	2123	–	4669	–	23	10301
		19	150	153	–	1.5	–	1951	–	1485	–	15	10800
		21	150	153	–	1.5	–	1800	–	2710	–	15	9728
		11	4016	4165	3908	3.7	−2.7*	1875	215	831	11	28	8221
		13	986	1019	980	3.3	−0.6*	2588	250	293	27	15	10617
	0.3	15	952	959	–	0.7	–	2297	–	7096	–	29	10445
		17	944	958	–	1.4	–	2204	–	7666	–	36	8445
		19	943	957	–	1.5	–	2064	–	2882	–	34	8644
		21	943	957	–	1.5	–	1910	–	278	–	6	7145
		11	2682	2697	–	0.5	–	2380	–	32	–	23	3713
		13	584	593	–	1.5	–	3341	–	2056	–	27	10800
*tw*2	0.6	15	570	575	–	1.0	–	2928	–	1178	–	17	7902
		17	556	560	–	0.7	–	2426	–	81	–	7	7032
		19	552	554	–	0.4	–	2076	–	18	–	33	7118
		21	546	550	–	0.8	–	2063	–	50	–	18	9275
		11	994	1128	1104	13.4	11.0	3419	89	1933	37	59	10694
		13	189	190	–	0.4	–	3856	–	219	–	28	10800
	0.9	15	170	171	–	0.4	–	3037	–	32	–	21	9188
		17	169	170	–	0.4	–	2412	–	113	–	15	8202
		19	169	170	–	0.4	–	2332	–	200	–	8	9372
		21	161	162	–	0.7	–	2292	–	1122	–	25	10201
		11	10332	10444	–	1.1	–	880	–	14	–	4	7185
		13	3126	3505	3072	12.1	−1.7*	2141	204	655	49	28	10578
	0.3	15	1561	1661	1643	6.4	5.2	2338	138	457	55	141	9183
		17	1105	1108	–	0.3	–	2361	–	12	–	9	4987
		19	1101	1107	–	0.6	–	2094	–	474	–	7	8079
		21	1076	1084	–	0.7	–	1973	–	838	–	7	3395
		11	7887	8365	7935	6.1	0.6	1484	308	98373	7	106	9903
		13	2854	2951	2790	3.4	−2.2*	2506	133	29733	19	117	9403
*tw*3	0.6	15	1159	1211	1140	4.5	−1.7*	2810	197	402	43	21	7522
		17	804	813	–	1.2	–	2724	–	68	–	13	4863
		19	764	769	–	0.6	–	2555	–	171	–	11	4488
		21	719	721	–	0.2	–	2077	–	44	–	8	3418
		11	6497	6839	6837	5.3	5.2	2069	404	142108	3	45	7660
		13	2447	2696	2354	10.1	−3.8*	3065	227	38014	43	68	10798
	0.9	15	959	961	–	0.3	–	3102	–	0	–	7	9515
		17	440	444	–	0.7	–	3052	–	9	–	24	8530
		19	397	398	–	0.2	–	2672	–	6	–	16	6062
		21	345	346	–	0.2	–	2253	–	62	–	8	2824

was used, fewer nodes were explored than those inspected by the B&B algorithm.

With regard to the number of columns generated, for B instances it suffices to generate around 500 columns at the root node of the B&B to solve the linear relaxation to optimality. In the larger AB instances around 2500 columns are required. When the B&P algorithm is used, the number of columns required does not seem to follow a clear pattern. Note that some values in column BP-LR (Table 5) are negative, reflecting the difference between the B&P and the linear relaxation at the root node. These correspond to cases where the column generation at the root-node LP was terminated before reaching optimality.

Recall that we run the B&P algorithm only if the difference between LR and BB is larger than 3%. This does not occur often, indicating that the observed gap between LR and BB is generally small. However, the solution of the linear relaxation at the root node in terms of θ is fractional in most cases.

To illustrate the behavior of the solutions, we consider two instances in Tables 6 and 7. The first case (instance 1: *tw*2, $\beta = 0.9$, $\#Tec = 6$) shows a considerable BB-LR gap (12.4%), justifying the application of the B&P procedure. In contrast, the second case (instance 2: *tw*3, $\beta = 0.3$, $\#Tec = 8$) yields a small BB-LR difference (0.3%), and the B&B algorithm obtains a close-to-optimal solution without having to explore nodes other than

Table 6
Solution instance 1.

Instance 1	c_r	θ_r	$c_r \cdot \theta_r$
r1	11.0	0.828	9.1
r2	12.5	0.237	3.0
r3	11.2	0.250	2.8
r4	12.5	0.113	1.4
r5	12.0	0.363	4.4
r6	11.2	0.250	2.8
r7	12.1	0.250	3.0
r8	12.4	0.207	2.6
r9	12.7	0.500	6.4
r10	12.0	0.137	1.6
r11	11.4	0.250	2.9
r12	11.6	0.650	7.5
r13	11.0	0.172	1.9
r14	12.5	0.150	1.9
r15	11.4	0.457	5.2
r16	11.4	0.293	3.3
r17	11.6	0.250	2.9
r18	11.4	0.250	2.9
r19	14.9	0.172	2.6
Other routes (~ 30)	28.4	< 0.100	2.8
Obj. LR			70.9
Obj. BB			79.7
Gap [%] BB-LR			12.4

Table 7
Solution instance 2.

Instance 2	c_r	θ_r	$c_r \cdot \theta_r$
r1	73.3	0.250	18.3
r2	70.9	0.750	53.2
r3	56.0	0.500	28.0
r4	71.4	0.250	17.9
r5	70.2	0.750	52.7
r6	73.2	0.250	18.3
r7	76.3	0.250	19.1
r8	68.2	0.250	17.1
r9	63.4	0.250	15.9
r10	60.7	0.250	15.2
r11	56.2	0.250	14.1
r12	78.8	0.250	19.7
r13	80.1	0.250	20.0
r14	74.9	0.250	18.7
r15	94.7	0.750	71.0
r16	82.7	0.250	20.7
r17	79.7	0.750	59.8
r18	57.4	0.500	28.7
r19	56.7	0.250	14.2
r20	82.0	0.250	20.5
r21	84.7	0.250	21.2
r22	99.4	0.250	24.9
Obj. LR			588.8
Obj. BB			590.4
Gap [%] BB-LR			0.3

the root. In the tables, we show the columns considered in the linear relaxation at the root-node solution in each case, with the corresponding $\theta > 0$ value associated with each column. Although both solutions are fractional, there are two aspects that could explain the difference in the BB-LR gap: the nature of the fractions (θ values) and the number of columns in the solution. The LR of instance 1 (large gap) results in a large range of θ values. However, instance 2 has only three θ values (0.25,0.5,0.75) in all the chosen columns. This occurred in most of the cases with a small BB-LR gap (see Tables 3–5). Moreover, the number of columns included in the linear relaxation at the root-node solution of instance 1 is much larger than the number for instance 2 (more than 50 compared with 22).

In summary, most of the solutions with a small BB-LR gap have a small number of columns in the linear relaxation at the root-node solution and a limited number of θ values (0.25,0.5,0.75). These examples show that when the BB-LR gap is large, the linear relaxation at the root-node solution has a range of fractional values (see instance 1) shared among a large number of columns.

6. Conclusions

In this paper, a technician dispatch problem is modeled as a vehicle routing problem with soft time windows. We solved several real examples provided by the operation of a large company in different service areas of Santiago, reformulating the problem using a set covering model and using constraint programming to solve the subproblems. Constraint programming allowed the subproblems to be solved easily and effectively.

Our model allows us to optimize the dispatching of technicians in two actual instances (areas A and B as shown in Fig. 4), showing the advantages of the proposed B&P algorithm over the manual solutions implemented by the company. In addition, we were able to run our model over the entire area AB (70 service requests), obtaining in some cases significant performance improvements by not restricting the solution to the prespecified subregions A and B.

Most instances had a small gap between the linear relaxation at the root node and the B&B solution, so it was not necessary to use B&P to explore beyond the root node. When B&P was used, it reduced the gap by 3.5% on average.

Acknowledgments

The authors wish to acknowledge the support of Fondecyt, Chile, Grants 1100239 and 1085188; the Millennium Institute Complex Engineering Systems (ICM: P-05-004-F, CONICYT: FBO16); and the Discovery Grant Program of the Canadian Natural Sciences and Engineering Research Council. The authors also thank Xerox Chile for providing the real problem that motivates this research as well as the data to develop the model along with the proposed solution method. Finally, the authors are grateful to several anonymous reviewers for their valuable comments.

References

Baldacci, R., Mingozzi, A., & Roberti, R. (2012). Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints. *European Journal of Operational Research, 218*, 1–6.

Barnhart, C., Hane, C., & Vance, P. (2000). Using branch and price and cut to solve origin–destination integer multicommodity flow problems. *Operations Research, 48*(2), 344–347.

Barnhart, C., Johnson, E. L., Nemhauser, G. L., Savelsbergh, M., & Vance, P. (1998). Branch-and-price: Column generation for solving huge integer programs. *Operations Research, 3*, 316–329.

Blakeley, F., Bozkaya, B., Cao, B. Y., Hall, W., & Knolmajer, J. (2003). Optimizing periodic maintenance operations for Schindler elevator corporation. *Interfaces, 33*, 67–79.

Bräysy, O., & Gendreau, M. (2005a). Vehicle routing problem with time windows, Part I: Route construction and local search algorithms. *Transportation Science, 39*(1), 104–118.

Bräysy, O., & Gendreau, M. (2005b). Vehicle routing problem with time windows, Part II: Meta-heuristics. *Transportation Science, 39*(1), 118–139.

Chabrier, A. (2006). Vehicle routing problem with elementary shortest path based column generation. *Computers & Operations Research, 33*(10), 2972–2990.

Cordeau, J., Desaulniers, G., Desrosiers, J., Solomon, M., & Soumis, F. (2002). *The vehicle routing problem.VRP with time windows* (pp. 157–193). SIAM.

Cordeau, J., Laporte, G., Pasin, F., & Ropke, S. (2010). Scheduling technicians and tasks in a telecommunications company. *Journal of Scheduling, 13*, 393–409.

Dantzig, G., & Wolfe, P. (1960). The decomposition algorithm for linear programming. *Operations Research, 8*, 101–111.

Desaulniers, G., Lessard, F., & Hadjar, A. (2008). Tabu search, partial elementarity, and generalized k-path inequalities for the vehicle routing problem with time windows. *Transportation Science, 42*(3), 387–404.

Desrochers, M., Desrosiers, J., & Solomon, M. (1992). A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research, 40*, 342–354.

Desrosiers, J., Solomon, M., & Soumis, F. (1995). *Handbooks in operations research and management science 8: Network routing. Time constrained routing and scheduling* (pp. 35–139). Elsevier.

Fahle, T., Junker, U., Karisch, S., Kohl, N., Sellmann, M., & Vaaben, B. (2002). Constraint programming based column generation for crew assignment. *Journal of Heuristics, 8*(1), 59–81.

Gualandi, S., & Malucelli, F. (2009). Constraint programming-based column generation. *4OR: A Quarterly Journal of Operations Research, 7*(2), 113–137.

Ibaraki, T., Imahori, S., Nonobe, K., Sobue, K., Uno, T., & Yagiura, M. (2008). An iterated local search algorithm for the vehicle routing problem with convex time penalty functions. *Discrete Applied Mathematics, 156*(11), 2050–2069.

Ilog. (2003a). ILOG CPLEX 9.0 User's Manual.

Ilog. (2003b). ILOG SOLVER 6.0 User's Manual.

Junker, U., Karisch, S., Kohl, N., Vaaben, B., 1999. Constraint programming based column generation for crew assignment. In *International workshop on integration of AI and OR techniques in constraint programming for combinatorial optimization problems (CP-AI-OR'99).*

Junker, U., Karisch, S., Kohl, N., Vaaben, B., Fahle, T., & Sellmann, M. (1999). A framework for constraint programming based column generation. *Lecture Notes in Computer Science*, 261–274.

Kovacs, A., Parragh, S., Doerner, K., & Hartl, R. (2012). Adaptive large neighborhood search for service technician routing and scheduling problems. *Journal of Scheduling, 15*(5), 579–600.

Liberatore, F., Righini, G., & Salani, M. (2011). A column generation algorithm for the vehicle routing problem with soft time windows. *4OR, 9*(1), 49–82.

Nagata, Y., Bräysy, O., & Dullaert, W. (2010). A penalty-based edge assembly memetic algorithm for the vehicle routing problem with time windows. *Computers & Operations Research, 37*(4), 724–737.

Pillac, V., Gueret, C., & Medaglia, A. L. (2013). A parallel matheuristic for the technician routing and scheduling problem. *Optimization Letters, 7*, 1525–1535.

Pisinger, D., & Ropke, S. (2007). A general heuristic for vehicle routing problems. *Computers & Operations Research, 34*(8), 2403–2435.

Régin, J.-C. (2011). Global constraints: A survey. In Milano, M., & Van Hentenryck, P. (Eds.), *Hybrid optimization. Springer optimization and its application* (Vol. 45, pp. 63–134).

Rousseau, L., Gendreau, M., & Feillet, D. (2007). Interior point stabilization for column generation. *Operations Research Letters, 35*(5), 660–668.

Rousseau, L., Gendreau, M., Pesant, G., & Focacci, F. (2004). Solving VRPTWs with constraint programming based column generation. *Annals of Operations Research, 130*(1), 199–216.

Rousseau, L., Pesant, G., & Gendreau, M. (2001). Building negative reduced cost paths using constraint programming. *Lecture Notes in Computer Science*. 778–778.

Ryan, D., & Foster, B. (1981). *Computer scheduling of public transport: Urban passenger vehicle and crew scheduling. Ch. An integer programming approach to scheduling* (pp. 269–280). Amsterdam: North-Holland.

Savelsbergh, M., & Sol, M. (1991). Drive: Dynamic routing of independent vehicles. *Operations Research, 46*(4), 474–490.

Shaw, P. (1998). Using constraint programming and local search methods to solve vehicle routing problems. *Principles and Practice of Constraint Programming – Cp98, 1520*, 417–431.

Souyris, S., Cortés, C. E., Ordoñez, F., & Weintraub, A. (2013). A robust optimization approach to dispatching technicians under stochastic service times. *Optimization Letters, 7*(7), 1549–1568.

Taillard, E., Badeau, P., Gendreau, M., Guertin, F., & Potvin, J. Y. (1997). A tabu search heuristic for the vehicle routing problem with soft time windows. *Transportation Science, 31*(2), 170–186.

Tang, H., Miller-Hooks, E., & Tomastik, R. (2007). Scheduling technicians for planned maintenance of geographically distributed equipment. *Transportation Research Part E–Logistics and Transportation Review, 43*, 591–609.

Vance, P., Atamturk, A., Barnhart, C., Gelman, E., Johnson, E., Krishna, A., et al. (1997). *A heuristic branch-and-price approach for the airline crew pairing problem. Tech. rep*. Georgia Tech, Supply Chain and Logistics Institute.

Vanderbeck, F. (2000). On Dantzig–Wolfe decomposition in integer programming and ways to perform branching in a branch-and-price algorithm. *Operations Research, 48*(1), 111–128.

Vanderbeck, F. (2005). *Column generation. Ch. Implementing mixed integer column generation*. Kluwer.

Vanderbeck, F., 2006. *Branching in branch-and-price: A generic scheme. Working paper. Applied mathematics U-05.14*. University Bordeaux.

Vanderbeck, F., & Savelsbergh, M. (2006). A generic view of Dantzig–Wolfe decomposition in mixed integer programming. *Operations Research Letters, 24*(3), 296–306.

Vidal, T., Crainic, T., Gendreau, M., & Prins, C. (2013). A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. *Computers & Operations Research, 40*, 475–489.

Weigel, D., & Cao, B. (1999). Applying GIS and OR techniques to solve Sears technician-dispatching and home-delivery problems. *Interfaces, 29*(1), 112–130.

Weintraub, A., Abud, J., Fernandez, C., Laporte, G., & Ramirez, E. (1999). An emergency vehicle dispatching system for an electric utility in Chile. *Journal of the Operational Research Society, 50*(7), 690–696.

Xu, J., & Chiu, S. Y. (2001). Effective heuristic procedures for a field technician scheduling problem. *Journal of Heuristics, 7*, 495–509.

Yunes, T., Moura, A., & de Souza, C. (2000). A hybrid approach for solving large scale crew scheduling problems. *Lecture Notes in Computer Science*, 293–307.

Yunes, T., Moura, A., & De Souza, C. (2005). Hybrid column generation approaches for urban transit crew management problems. *Transportation Science, 39*(2), 273–288.