An Exact Algorithm for the Elementary Shortest Path Problem with Resource Constraints: Application to some Vehicle Routing Problems

Dominique Feillet[1]
Pierre Dejax[2]
Michel Gendreau[3]
Cyrille Gueguen[4]

[1] Laboratoire d'Informatique d'Avignon
Université d'Avignon
339 Chemin des Meinajariés, BP 1228, 84911 Avignon Cedex 9, France
dominique.feillet@lia.univ-avignon.fr

[2] Département automatique et Productique
Ecole des mines de Nantes
4, rue Alfred Kastler, BP20722, 44307 Nantes Cedex 3, France
Pierre.Dejax@emn.fr

[3] Centre de recherche sur les transports
Université de Montréal
C.P. 6128 succursale Centre-ville, Montréal, Canada H3C 3J7
michelg@crt.umontreal.ca

[4] Département de recherche opérationnelle
Direction Générale des Systèmes d'Information, AIR FRANCE
1, av. du Maréchal Devaux, 91551 Paray-Vieille-Poste Cedex, France
cygueguen@airfrance.fr

Abstract In this paper, we propose a solution procedure for the Elementary Shortest Path Problem with Resource Constraints (ESPPRC). A relaxed version of this problem in which the path does not have to be elementary has been the backbone of a number of solution procedures based on column generation for several important problems, such as vehicle routing and crew-pairing. In many cases relaxing the restriction of an elementary path resulted in optimal solutions in a reasonable computation time. However, for a number of other problems, the elementary path restriction has too much impact on the solution to be relaxed or might even be necessary. We propose an exact solution procedure for the ESPPRC which extends the classical label correcting algorithm originally developed for the relaxed (non-elementary) path version of this problem. We present computational experiments of this algorithm for our specific problem and embedded in a column generation scheme for the classical Vehicle Routing Problem with Time Windows.

Key words: Shortest path, column generation, vehicle routing.

Résumé Dans cet article nous proposons une procédure de résolution du Problème du Plus Court Chemin Elémentaire avec Contraintes de Ressources (ESPPRC). Une version relaxée de ce problème, dans laquelle le chemin n'a pas à être élémentaire, a été l'épine dorsale de nombreuses procédures de résolutions basées sur les techniques de génération de colonnes, notamment pour des problèmes de tournées de véhicules ou d'horaires d'équipes de travail. Dans de nombreux cas, cette relaxation permettait d'obtenir des solutions optimales dans des temps raisonnables. Cependant, pour un certain nombre d'autres problèmes, il est nécessaire ou important pour l'efficacité de la résolution de maintenir les plus courts chemins élémentaires. Nous proposons une procédure de résolution exacte pour le ESPPRC qui étend l'algorithme classique de programmation dynamique développé dans le cas de chemins non élémentaires. Nous présentons des expérimentations numériques pour le problème propre, ainsi que pour un algorithme de génération de colonnes faisant appel à lui pour résoudre le Problème de Tournées de Véhicules avec Fenêtres de Temps.

Mots-clés : plus court chemin, génération de colonnes, tournées de véhicules.

1 Introduction

In this paper, we propose a solution procedure for the Elementary Shortest Path Problem with Resource Constraints (ESPPRC). A relaxed version of this problem, the Shortest Path Problem with Resource Constraints (SPPRC), in which the path does not have to be elementary, has been the backbone of a number of solution procedures based on column generation. This approach has successfully been applied by Desrochers *et al.* [8] for the Vehicle Routing Problem with Time Windows (VRPTW), by Lavoie *et al.* [23] for the airline crew pairing, by Graves *et al.* [14] for the flight crew scheduling to mention only some examples. In these problems, one might have resource constraints related to time (time windows, route duration), capacity (vehicle capacity) and many other types of resources. For these problems, the column generation approach involving the iterative solution of SPPRC instances results in optimal solutions in a reasonable computation time.

The standard approach to solve the SPPRC in practice is based on dynamic programming and has a pseudo-polynomial complexity. Briefly, the principle is to associate with each possible partial path a label indicating the consumption of resources and to eliminate labels with the help of dominance rules. In label correcting approaches, nodes are repeatedly treated and their labels extended. This approach is an extension of the Ford-Bellman algorithm. It was proposed by Desrochers [7] in the SPPRC context. Within this approach, two strategies can be employed. In reaching algorithms, labels on a node are extended to its successors. In pulling algorithms, labels from its predecessors are pulled to the node currently treated. Besides the label correcting approach, the label setting approach is an extension of Dijkstra's algorithm and works by the permanent marking of the labels, that are treated in an order based on the resource consumption.

Another important solution approach for the SPPRC stems from the integer linear programming formulation of the problem and the use of a Lagrangean relaxation technique. This approach takes advantage of the effectiveness of algorithms that solve the unconstrained shortest path problem. Handler and Zang [16] and Beasley and Christofides [2] address the problem this way, as Borndörfer *et al.* [3] or Ziegelmann [26] more recently. Note that Ziegelmann [26] also provides a survey devoted to the SPPRC, where more detailed information is available and other solution approaches are mentioned.

Solving the SPPRC with a dynamic programming approach is also closely related to solving multi-objective shortest path problems, which have received the attention of many researchers. Indeed, the aim in these problems is also to generate non-dominated paths (i.e., Pareto optimal paths). However, the methods developed in the literature usually concern multi-objective shortest path problems on graphs with non-negative lengths (see, for instance, Warburton [25] and Hansen *et al.* [17]). The interested reader may find an exhaustive review of multi-objective shortest path problems in Current and Marsh [5].

Unlike the SPPRC, the ESPPRC has been relatively neglected up to now. However, for some vehicle routing applications that we will mention later, the subproblem encountered in the column generation procedure would advantageously be solved as an ESPPRC. Dror [9] proved that the ESPPRC is NP-hard in the strong sense. To our knowledge, no paper has been dedicated to its solution when the graph contains negative costs, which frequently happens in the context of column generation. Gondran and Minoux [13] did, however, propose a sub-optimal approach for solving the Elementary Shortest Path Problem (without resource constraint) in graphs that might contain negative cost cycles.

The algorithm proposed in this paper is able to solve optimally the ESPPRC in general graphs, even if they contain negative cost cycles. In Section 2, we describe how the column generation technique is applied to the VRPTW by Desrochers *et al.* [8]. We then highlight the advantages of maintaining the elementary path condition within the column generation process. Section 3 describes formally the ESPPRC. In Section 4, we extend Desrochers' [7] label correcting algorithm for the solution of the ESPPRC. Computational results of Section 5 conclude the paper.

2 Shortcomings of the classical column generation solution approach for some vehicle routing problems

The basic VRPTW consists in designing an optimal set of delivery or collection routes from a depot to a number of customers, subject to time windows and capacity constraints. One of the most successful implementations of the column generation approach for its solution has been proposed by Desrochers *et al.* [8]. The backbone of this approach is the introduction of the SPPRC as a subproblem within the column generation procedure. Desrochers *et al.* [8] first formulate the VRPTW using the Classical Set Partitioning Model:

$$\text{minimize} \sum_{r_k \in \Omega} c_k x_k \tag{1}$$

subject to

$$\sum_{r_k \in \Omega} a_{ik} x_k = 1 \qquad \forall v_i \in V, \tag{2}$$

$$x_k \in \{0,1\} \qquad \forall r_k \in \Omega, \tag{3}$$

where $V = \{v_1, \ldots, v_n\}$ is the set of customers, Ω is the set of feasible elementary routes, c_k is the cost of route $r_k \in \Omega$ and a_{ik} is equal to 1 if route $r_k \in \Omega$ visits customer v_i and 0 otherwise. This model suggests formulating the subproblem as an ESPPRC.

Instead of the set partitioning model described above, Desrochers *et al.* [8] introduce a set covering formulation, where they replace constraints (2) with constraints

$$\sum_{r_k \in \Omega} a_{ik} x_k \geq 1 \qquad \forall v_i \in V \tag{4}$$

and where a_{ik} now represents the number of times route $r_k \in \Omega$ visits customer v_i. This formulation no longer requires that the paths be elementary (the set Ω is extended) and the subproblem changes to an SPPRC. Moreover, this relaxation does not invalidate the solution process provided that the triangle inequality is satisfied because, in this case, there exists an optimal solution of the integer program that visits each customer at most once.

The approach of Desrochers *et al.* [8] is computationally very attractive because the SPPRC can be solved efficiently with dynamic programming. However, this method has a significant disadvantage, which is a weakening of the lower bound, as mentioned, for example, by Gelinas *et al.* [12]. The computational results of Section 5.2 will permit us to assess this weakening of the lower bound for the VRPTW. Apart from this work, we also tested column generation for other classes of routing problems where the relaxation was even more detrimental. An illustrative example is the Vehicle Routing Problems with Profits class (see Feillet *et al.* [11]), where it is not necessary to visit all customers, but profit is collected at visited nodes. For this class of problems, the relaxation of the elementary path restriction induces the collection of profit each time a customer is visited; this leads to a very bad evaluation of the quality of the routes and, consequently, a very bad bound.

In the context of the VRPTW, some work has been carried out in order to limit the weakening of the lower bound, with the help of valid inequalities (Kohl *et al.* [21]) or by eliminating cycles that contain no more than 2 arcs (e.g., Desrochers *et al.* [8]) or no more than k arcs (Irnich [19]). Even so, the potential for lower bound improvement justifies the interest of developing an exact method for solving the ESPPRC.

Note furthermore that the relaxation of the elementary path condition could induce a malfunction of the algorithm, depending on the problem and on the branching scheme. For Vehicle Routing Problems with Profits, optimal elementary routes might be dominated by non-elementary routes, as soon as the branching scheme does not prohibit cycles. In other problems, where vertices might be visited more than once (e.g., the Split Delivery VRP or the Capacitated Arc Routing Problem), the relaxation could lead to the appearance of negative cost cycles that do not consume resources, which would disrupt the proper functioning of the algorithm.

Before describing our solution procedure for the ESPPRC, we first state the problem formally.

3 Description of the ESPPRC

Let $G = (V, A)$ be a network, where A is the set of arcs and $V = \{v_1, \ldots, v_n\}$ is the set of nodes, including an origin node p and a destination node d. A cost c_{ij} is associated with each arc $(v_i, v_j) \in A$. Let L be the number of resources and $d_{ij}^l \geq 0$ be the consumption of resource l along arc (v_i, v_j). Values d_{ij}^l are assumed to satisfy the triangle inequality for each resource l. With each node v_i and each resource l are associated two nonnegative values a_i^l and b_i^l, such that the consumption of resource l along a path from p to v_i is constrained to belong to the interval $[a_i^l, b_i^l]$. If the consumption of resource l is lower than a_i^l when the path reaches v_i, it is set to a_i^l. Note that this notation is natural for the time resource, but also allows us to represent capacity constraints by defining intervals $[0, Q]$ on nodes, where Q is the capacity limit. The objective is to generate a minimum cost elementary path from p to d that satisfies all resource constraints.

We now describe an algorithm for solving the ESPPRC in general graphs, even if they contain negative cost cycles.

4 A label correcting algorithm to solve the ESPPRC

Let us first say that adjusting Desrochers' label correcting algorithm [7] in order to solve the ESPPRC instead of the SPPRC does not seem trivial. Indeed, it is easy to notice that one cannot find an optimal elementary path by just solving an SPPRC and selecting an elementary path among the set of optimal paths. It is also obvious that one cannot find an optimal elementary path by solving the SPPRC and by enforcing that only elementary paths be generated at each step of the process. As a matter of fact, one should not expect to solve a strongly NP-hard problem using a pseudo-polynomial algorithm.

While proposing a solution procedure for the SPPRC, Beasley and Christofides [2] described how to find elementary paths in a graph using their algorithm: one only has to add an extra binary resource for each node $v_k \in V$ that is consumed when the node is visited. Beasley and Christofides [2] thought that their approach would only be suitable for solving problems with a small number of resources and they did not investigate the method further.

In this paper, we adapt this idea to Desrochers' label correcting algorithm [7]and strengthen it. Note that we do not investigate the possibilities of developing an algorithm with a different framework (as a label setting algorithm for example), since our purpose was to focus on the elementary path question. In the following, we use Desrochers' notation, which we recall below.

4.1 Notation and principle of Desrochers' algorithm for the SPPRC

With each path X_{pi} from the origin node p to a node v_i are associated a state $(T_i^1, T_i^2, \ldots, T_i^L)$ corresponding to the quantity of each of the L resources used by the path and a cost $C(T_i^1, T_i^2, \ldots, T_i^L)$. To simplify the notation, a label (R_i, C_i) is defined for each path X_{pi}, where $R_i = (T_i^1, T_i^2, \ldots, T_i^L)$ and $C_i = C(T_i^1, T_i^2, \ldots, T_i^L)$.

Let X'_{pi} and X^*_{pi} be two distinct paths from p to v_i with associated labels (R'_i, C'_i) and (R^*_i, C^*_i) respectively, X'_{pi} dominates X^*_{pi} if and only if $C'_i \leq C^*_i$, $T_i'^l \leq T_i^{*l}$ for $l = 1, \ldots, L$ and $(R'_i, C'_i) \neq (R^*_i, C^*_i)$.

To obtain the optimal solution of the shortest path problem, one just needs to consider non-dominated labels, i.e., non-dominated paths (Desrochers [7]).

4.2 New definitions and claims

As proposed by Beasley and Christofides [2], we change the label definition and add n extra binary resources, one for each node $v_k \in V$. In order to compare labels more efficiently, we also add another resource. This resource s_i counts the number of nodes visited by a path X_{pi}.

Definition 1 *With each path X_{pi} from the origin node p to a node v_i, associate a state $R_i = (T_i^1, \ldots, T_i^L, s_i, V_i^1, \ldots, V_i^n)$ corresponding to the quantity of each resource used by the path, the number of visited nodes, and the visitation vector ($V_i^k = 1$ if the path visits node v_k, 0 otherwise).*

With this new definition of labels, the dominance relation in the comparison of two labels is now stated as:

Definition 2 *Let X'_{pi} and X^*_{pi} be two distinct paths from p to v_i with associated labels (R'_i, C'_i) and (R^*_i, C^*_i). X'_{pi} dominates X^*_{pi} if and only if $C'_i \leq C^*_i$, $s'_i \leq s^*_i$, $T_i'^l \leq T_i^{*l}$ for $l = 1, \ldots, L$, $V_i'^k \leq V_i^{*k}$ for $k = 1, \ldots, n$, and $(R'_i, C'_i) \neq (R^*_i, C^*_i)$.*

This dominance relation is actually a simple restatement of the relation proposed by Desrochers [7]. Note that $s_i = \sum_{k=1}^{n} V_i^k$ and that this resource is only needed for computational purposes, since a label cannot dominate another label if it visits more nodes.

These modifications still allow the use of the original Desrochers' algorithm. The principle of the algorithm remains the same: All non-dominated paths are the extension of a non-dominated path. Indeed, the extension of a dominated path X_{pi} with an arc (v_i, v_j) results in a path that is either dominated by or equal to the extension X'_{pj} of the path X'_{pi} that dominates X_{pi}. Thus, during the execution of the algorithm, one only needs to consider non-dominated paths.

This new definition of labels ensures we generate only labels corresponding to elementary paths and we get the optimal solution of the problem. However, the size of the state space increases drastically and one is likely to generate and to keep many more labels during the solution.

4.3 Improved definition of the labels

In this section, we once more propose another definition of the labels, in order to limit their number throughout the process. Let us consider a partial path and its associated label. In the previous definition, the nodes belonging to the path were stored in the visitation vector of the label. As these nodes have already been visited, they cannot be visited in any extension of the corresponding path. In addition, there may be other nodes which cannot be visited in any extension of the corresponding path, due to resource limitations. The principle of this new definition is that it is more efficient to know which nodes cannot be visited anymore, whatever the reason. We call such nodes unreachable. When a label is extended, visitation resources corresponding to nodes that cannot be visited anymore (either because they have already been visited or because of resource constraints) are consumed. This algorithmic modification is computationally attractive because the dominance relation becomes sharper, as is shown in the following example.

Let us consider an instance of the ESPPRC and focus on two nodes v_1, v_2 and two labels $\lambda_2^a = (10, s_2^a, 0, 1, C_2^a)$ and $\lambda_2^b = (5, s_2^b, 1, 1, C_2^b)$ on node v_2. The resources respectively correspond to time, the number of visited nodes and the visitation vector restricted to $\{v_1, v_2\}$. We suppose that $s_2^b \leq s_2^a$, that $C_2^b \leq C_2^a$ and that $V_2^{k\,b} \leq V_2^{k\,a}$ for all the other nodes of the graph. Let us finally suppose that label λ_2^a cannot be extended to node v_1 due to the time resource limitation. With the dominance relation of Section 4.2, neither of these two labels dominates the other. Yet we are sure that any extension of λ_2^a until the destination can be replicated for λ_2^b, and that the path resulting from the extension of λ_2^a will be more costly. Using the approach described in the previous paragraph, the labels become $\lambda_2'^{\,a} = (10, s_2'^{\,a}, 1, 1, C_2^a)$ and $\lambda_2'^{\,b} = (5, s_2'^{\,b}, 1, 1, C_2^b)$, where the resources are respectively time, the number of unreachable nodes and the vector of unreachable nodes. With this new definition, $\lambda_2'^{\,b}$ dominates $\lambda_2'^{\,a}$, which was expected.

We now have to formalise these ideas. Let us define the concept of unreachable node and propose a new definition for the labels.

Definition 3 *For each path X_{pi} from the origin node p to a node $v_i \in V$, a node v_k is said to be unreachable if it is included in X_{pi} or if there exists a resource $l \in \{1, \ldots, L\}$ satisfying $T_i^l + d_{ik}^l > b_k^l$ (which means that the current value of consumption of l prevents the path from reaching node v_k).*

Definition 4 *With each path X_{pi} from the origin node p to a node $v_i \in V$, associate a state $R_i = (T_i^1, \ldots, T_i^L, s_i, V_i^1, \ldots, V_i^n)$ corresponding to the quantity*

of the resources used by the path, the number of unreachable nodes and the vector of unreachable nodes, defined by $V_i^k = 1$ *if node* v_k *is unreachable.*

Note that a node is said to be unreachable when it cannot be attained directly using an outgoing arc. It also means that there does not exist any path that permits to reach it since the triangle inequality holds for the resources. Note also that we still have $s_i = \sum_{k=1}^{n} V_i^k$.

With the above definitions, we can keep the same dominance relation and only consider non-dominated paths.

Claim 1 *During the execution of the modified algorithm, we need only to consider non-dominated paths.*

Proof

Consider two labels on node v_i, (R'_i, C'_i) and (R_i, C_i) such that (R'_i, C'_i) dominates (R_i, C_i). Let X'_{pi} and X_{pi} be the respective associated paths from the origin node p to node v_i and consider an arc (v_i, v_j) such that the extension of X_{pi} by (v_i, v_j) is a feasible elementary path from p to v_j. Denote this path X_{pj}. We have to show that the extension of X'_{pi} by (v_i, v_j) also leads to a feasible elementary path X'_{pj} from p to v_j that is either equal to or dominates X_{pj}. The claim follows by applying this result inductively.

The first point is to show that X'_{pi} can be extended by (v_i, v_j). $T_i'^l \leq T_i^l$ for $l = 1, \ldots, L$. Since $T_j'^l = max\{a_j^l, T_i'^l + d_{ij}^l\}$ and $T_j^l = max\{a_j^l, T_i^l + d_{ij}^l\}$, we have $T_j'^l \leq T_j^l$ for $l = 1, \ldots, L$. We also have $V_i'^j \leq V_i^j$ and $V_i^j = 0$, since (R'_i, C'_i) dominates (R_i, C_i) and since it is possible to extend X_{pi} by (v_i, v_j). Thus, $V_i'^j = 0$ and it is possible to extend X'_{pi} by (v_i, v_j).

We now have to show that X'_{pj} is either equals to or dominates X_{pj}. We already know that $T_j'^l \leq T_j^l$ for $l = 1, \ldots, L$. The definition of the vector of unreachable nodes implies that $V_j'^j = V_j^j = 1$. It is also clear that $C'_j = C'_i + c_{ij} \leq C_i + c_{ij} = C_j$. Thus, it just remains to check that $V_j'^k \leq V_j^k$ for every node $v_k \neq v_j$ and that $s'_j \leq s_j$. Let us begin with the vector of unreachable nodes. We have to show that a node that is unreachable for X'_{pj} is also unreachable for X_{pj}. Let $v_k \neq v_j$ be an unreachable node for X'_{pj}. Using the definition of unreachable nodes, v_k is either included in X'_{pj} or there exists a resource $l \in \{1, \ldots, L\}$ satisfying $T_j'^l + d_{jk}^l > b_k^l$. But:

- First, if v_k is included in X'_{pj}, it is included in X'_{pi} and $V_i'^k = 1$. We also know that $V_i'^k \leq V_i^k$ and that $V_i^k \leq V_j^k$. This permits to conclude that $V_j^k = 1$.
- Second, if there exists $l \in \{1, \ldots, L\}$ with $T_j'^l + d_{jk}^l > b_k^l$, $T_j{}^l + d_{jk}^l > b_k^l$ since $T_j'^l \leq T_j^l$ for $l = 1, \ldots, L$.

Thus, if a node $v_k \neq v_j$ is unreachable for X'_{pj}, it is also unreachable for X_{pj}, which means exactly that $V_j'^k \leq V_j^k$ for every node $v_k \neq v_j$. Finally, this last result implies that $s'_j \leq s_j$, from which we conclude that (R'_j, C'_j) is either equal to or dominates (R_j, C_j) $\square$

It is easy to implement this label modification. Indeed, while extending a label, we just have to update the set of unreachable nodes. For this purpose, we assess the feasibility of an extension through every outgoing arc. Note that the computation time devoted to this task depends on the value of L. When L is too large, the definition of an unreachable node could be adapted by only taking a subset of prominent resources into account, in order to limit the computation times. Otherwise, when L is low, it does not take much longer to determine the unreachable nodes than to store the nodes that have been visited. A complete description of the algorithm is given below.

4.4 Description of the algorithm

We need the following notation to describe the algorithm:

- Λ_i: List of labels on node v_i.
- $Succ(v_i)$: Set of successors of node v_i.
- E: List of nodes waiting to be treated.
- $Extend(\lambda_i, v_j)$: Function that returns the label resulting from the extension of label $\lambda_i \in \Lambda_i$ towards node v_j when the extension is possible, nothing otherwise. The function first updates the consumption of resources $l = 1, \ldots, L$. If the resource constraints are satisfied, it explores the set of outgoing arcs in order to update the vector of unreachable nodes and the number of unreachable nodes.
- F_{ij}: Set of labels extended from node v_i to node v_j.
- $EFF(\Lambda)$: Procedure that keeps only non-dominated labels in the list of labels Λ.

We now describe the procedure ESPPRC(p) which determine all the non-dominated paths starting from node p to every node of the graph.

ESPPRC(p)
1 INITIALIZATION
2 $\Lambda_p \leftarrow \{(0, ..., 0)\}$
3 **for all** $v_i \in V \setminus \{p\}$
4 **do** $\Lambda_i \leftarrow \emptyset$

```
5   E = {p}
6
7   repeat
8           Exploration of the successors of a node
9           Choose v_i ∈ E
10          for all v_j ∈ Succ(v_i)
11              do F_ij ← ∅
12                  for all λ_i = (T_i^1, ..., T_i^L, s_i, V_i^1, ..., V_i^n, C_i) ∈ Λ_i
13                      do if V_i^j = 0
14                              then F_ij ← F_ij ∪ {Extend (λ_i, v_j)}
15                  Λ_j ← EFF (F_ij ∪ Λ_j)
16                  if Λ_j has changed
17                    then E ← E ∪ {v_j}
18          Reduction of E
19          E ← E\{v_i}
20    until E = ∅
```

The time complexity of this algorithm is strongly related to the structure of the graph, the numbering of the nodes and the tightness of resource constraints. If the problem is highly constrained, it is possible to solve quite large problems as it is shown in Section 5.1.

5 Computational results

The computational study is divided into two parts. First, we present experiments on ESPPRC instances. Second, we tackle the solution of VRPTW instances with a column generation approach using the ESPPRC as a subproblem. Let us recall that the VRPTW does not require the use of an ESPPRC solution algorithm, but our purpose was to highlight the differences with the usual solution methodology. For both parts, we use Solomon's data sets [24]. These data sets are classified in three categories: The r-instances where the customers are located randomly, the c-instances where the customers are located in clusters and the rc-instances with some random and some clustered structures. The number of customers is 100 for every instance but smaller instances are created by considering only the first 25 or the first 50 customers.

Each instance is represented by a complete graph $G = (V, A)$, where $V = \{v_1, \ldots, v_n\}$ is the set of customers. In the data sets, the distance matrix is not explicitly stated, but customer locations are given. We note $dist_{ij}$ the distance between two customers v_i and v_j and we define $dist_{ij}$ as the Euclidean distance between these customers, calculated with one decimal point and truncation. It is interesting to note that working with non-truncated values would not have affected our computation times. Two resources are defined: time and load. Time

consumption d^1_{ij} is given by $d^1_{ij} = st_i + dist_{ij}$ where v_i and v_j are two customers and st_i represents the service time for customer v_i. Load consumption d^2_{ij} is given by $d^2_{ij} = q_j$ where q_j is the quantity to deliver to customer v_j. Each customer v_i receives a time interval $[a^1_i, b^1_i]$ representing the time window and a load interval $[0, Q]$ representing the load limit, where Q is the vehicle capacity. Due to the resource constraints, and especially to the time windows in the present case, some connections are forbidden. The corresponding arcs are then removed from the graph.

Computational experiments were carried out on a PC with a 333 MHz processor and with 128 Megabytes of RAM. Algorithms were implemented in C++ with Visual C++ 5.0 Compiler. The computation times were calculated with the *clock()* function. The maximum allowed time to find a solution was set to 300 seconds in the first part of the computational study and to 3600 seconds in the second part.

5.1 Using the algorithm to solve the ESPPRC

The efficiency of our algorithm is assessed on Solomon's data sets [24] with the following cost definition. With each arc $(v_i, v_j) \in A$ is associated a cost $c_{ij} = dist_{ij} - \alpha_i$, where α_i is a random integer variable, uniformly distributed in $\{0, \ldots, 20\}$. The limit value 20 has been chosen to generate a reasonable number of arcs having a negative cost. For each instance, we look for the shortest elementary path starting from the depot, coming back to an artificial copy of the depot and satisfying resource constraints.

Results are summarized in Tables 1, 2 and 3. For each instance, we first give the number of arcs in the graph, its density and the percentage of arcs with a negative cost. Column $\sharp$ *labels* gives the cardinality of the solution set (containing all the Pareto optimal paths), i.e., the number of labels on the destination node at the end of the algorithm. The computation time (in seconds) needed for the solution is given in column *CPU*. The last column gives the ratio of the computation times using or not the improvement described in Section 4.3. Empty rows or cells indicate that the problem was not solved within the 300s time limit allowed.

Our algorithm succeeds in solving 75 instances out of 87 in less than 300 seconds while the version without the improvement described in Section 4.3 permits only to solve 47 instances. Furthermore, Table 4 highlights the significant reduction of computing times resulting from this improvement. Note that the values of this table are obtained by considering only the 47 instances solved by both algorithms in the allowed time. It is also interesting to note that if we only consider instances solved in more than 10 seconds by the initial algorithm (before the improvement), which implies that the side effects of the data treatment disappear, ratios of Table 4 respectively increase up to 28.65 for the r-instances, 636.14 for the c-instances and 32.60 for the rc-instances.

Problem	♯ Arcs	Density	% Neg Arcs	Before Imp		After Imp		Ratio
				CPU	♯ labels	CPU	♯ labels	
r101.25	225	34.61	6.7	0.01	28	0.01	12	1
r101.50	813	31.88	8.0	0.04	83	0.03	11	1.3
r101.100	3244	32.12	9.9	11.29	2606	0.41	29	27.5
r102.25	413	63.54	7.7	0.01	37	0.02	22	0.5
r102.50	1504	58.98	7.0	0.68	317	0.22	42	3.1
r102.100	5813	57.55	9.6			280.35	1472	
r103.25	525	80.77	6.5	0.03	38	0.03	26	1
r103.50	1970	77.25	6.5	9.16	1221	1.01	101	9.1
r103.100	7705	76.29	8.7					
r104.25	580	89.23	6.2	0.03	41	0.05	32	0.6
r104.50	2398	94.40	5.9			58.96	1121	
r104.100	9187	90.96	8.1					
r105.25	301	46.31	6.0	0.01	301	0.01	12	1
r105.50	1068	41.88	7.4	0.18	146	0.09	17	2
r105.100	4261	42.19	9.7			1.63	84	
r106.25	462	71.08	7.1	0.02	462	0.03	22	0.7
r106.50	1696	66.51	6.8	3.68	832	0.49	84	7.5
r106.100	6559	64.94	9.5					
r107.25	550	84.61	6.2	0.03	40	0.04	26	0.7
r107.50	2092	82.04	6.1	38.43	2714	1.86	163	20.7
r107.100	8206	81.25	8.7					
r108.25	598	92.00	6.0	0.04	598	0.05	33	0.8
r108.50	2437	95.57	5.9			67	1196	
r108.100	9411	93.18	8.2					
r109.25	411	63.23	6.3	0.02	35	0.02	16	1
r109.50	1534	60.16	6.9	1.20	362	0.29	35	4.1
r109.100	6033	59.73	9.6			11.3	234	
r110.25	527	81.08	5.7	0.04	42	0.04	22	1
r110.50	2019	79.18	6.3	54.48	2919	1.17	119	46.6
r110.100	7966	78.87	8.9					
r111.25	526	80.92	6.5	0.03	39	0.03	26	1
r111.50	2035	79.80	6.4	29.97	2356	1.51	125	19.8
r111.100	7958	78.79	8.6					
r112.25	647	99.54	5.7	0.08	45	0.07	41	1.1
r112.50	2528	99.14	5.7			5.24	305	
r112.100	10014	99.15	8.0					

Table 1: Solution of the ESPPRC for r-instances.

Problem	♯ Arcs	Density	% Neg Arcs	Before Imp		After Imp		Ratio
				CPU	♯ labels	CPU	♯ labels	
c101.25	333	51.23	23	2.08	2206	0.02	51	104
c101.50	1218	47.76	14	154.54	9606	0.13	49	1188
c101.100	4516	44.71	8.3			0.86	75	
c102.25	475	73.08	24	88.94	6357	0.88	498	101.1
c102.50	1749	68.59	14			1.85	185	
c102.100	6684	66.18	7.5			99.04	456	
c103.25	570	87.69	21			6.46	560	
c103.50	2188	85.80	14			96.42	1301	
c103.100	8526	84.42	7.4					
c104.25	605	93.08	21			186.13	1837	
c104.50	2462	96.55	14					
c104.100	9597	94.96	7.3					
c105.25	358	55.08	22	2.31	2206	0.03	52	77
c105.50	1334	52.31	14			0.18	50	
c105.100	5052	50.02	7.6			1.15	77	
c106.25	337	51.85	23	2.06	2203	0.03	50	68.7
c106.50	1273	49.92	15	255.69	10625	0.16	56	1598.1
c106.100	5423	53.69	7.8			1.7	101	
c107.25	378	58.15	21	4.02	2698	0.03	58	134
c107.50	1444	56.63	13			0.22	65	
c107.100	6694	66.28	46			125.47	260	
c108.25	432	66.46	22	10.36	2816	0.08	85	129.5
c108.50	1643	64.43	14			0.45	102	
c108.100	6343	62.80	7.5			3.31	168	
c109.25	483	74.31	22	31.17	4134	0.19	138	164.0
c109.50	1896	74.35	14			1.5	227	
c109.100	7455	73.81	7.2			11.36	470	

Table 2: Solution of the ESPPRC for c-instances.

Problem	♯ Arcs	Density	% Neg Arcs	Before Imp		After Imp		Ratio
				CPU	♯ labels	CPU	♯ labels	
rc101.25	277	42.62	19	0.03	147	0.02	18	1.5
rc101.50	833	32.67	19	0.68	748	0.08	29	8.5
rc101.100	3642	36.06	7.9	18.30	2584	0.74	35	24.7
rc102.25	440	67.69	18	0.14	237	0.05	54	2.8
rc102.50	1463	57.37	14	10.74	1390	0.34	27	31.6
rc102.100	6002	59.43	6.5			8.96	148	
rc103.25	546	84.00	16	0.2	257	0.11	74	1.8
rc103.50	1941	76.12	13	66.44	1842	1.29	43	51.5
rc103.100	7820	77.43	5.8			242.92	1258	
rc104.25	592	91.08	16	0.67	312	0.23	116	2.9
rc104.50	2376	93.18	12			12.30	409	
rc104.100	9274	91.82	5.4					
rc105.25	371	57.08	20	0.06	185	0.03	22	2
rc105.50	1242	48.71	16	3.09	1155	0.28	28	11
rc105.100	5250	51.98	6.9	248.68	8700	8.54	75	29.1
rc106.25	388	59.69	19	0.08	277	0.04	34	2
rc106.50	1243	48.74	18	5.03	1606	0.33	31	15.2
rc106.100	5390	53.37	7.3			3.44	65	
rc107.25	514	79.08	18	0.44	278	0.14	45	3.1
rc107.50	1835	71.96	15	65.52	3232	2.51	111	26.1
rc107.100	7542	74.67	6.4			26.97	177	
rc108.25	617	94.92	16	1.01	307	0.34	92	3
rc108.50	2317	90.86	12			8.79	184	
rc108.100	9313	92.21	5.5			207.08	486	

Table 3: Solution of the ESPPRC for rc-instances.

Instance type	Before Imp CPU1	After Imp CPU2	Ratio
r-instances	6.79	0.34	6.92
c-instances	61.24	0.17	396.1
rc-instances	26.32	0.94	13.6

Table 4: Summary of average computation times and ratios (on the 47 instances solved by both algorithms).

Finally, these results tend to confirm the intuitive idea that the more negative arcs there are, the more complicated the problem is.

5.2 Using the algorithm inside a column generation solution procedure

Even if our algorithm can specifically be used for the solution of the ESPPRC, our major concern is its potential for the solution of complex routing problems using a column generation approach. In this section, we assess the algorithm efficiency on Solomon's instances [24], embedded in a column generation procedure devoted to the solution of the VRPTW. Because our objective is just to test the shortest path algorithm, i.e., the subproblem of the column generation scheme, we only address the solution of the linear relaxation of the problem. Even so, we directly obtain some integer solutions that are labeled with an asterisk in the result tables.

Linear programs are solved with CPLEX 6.0 [18]. The interface between CPLEX and the programming language is realized with the help of LPToolKit 3.6 [10].

In the following tables, we compare the results obtained with two methods. The first one is the usual column generation approach, where the subproblem is a SPPRC. In the second one, the elementary path restriction is not relaxed and we use our algorithm (see Section 2). For each instance and each method, we indicate the number of subproblems solved during the column generation process ($\sharp$ *iter*), the number of columns generated ($\sharp$ *col*), the linear relaxation value (LB) and the computation time (CPU). The column *Gap* gives the gap between the two relaxed values, i.e., the difference between the two lower bounds as a percentage of the elementary lower bound. The last column is the ratio $CPU2/CPU1$ when both methods have finished within the time limit of 3600 seconds. Cells are empty when results are not available (due to the time limit). Results are respectively presented in Tables 5, 6 and 7, for r-, c- and rc-instances.

Note that the column generation process is initiated with an adaptation of the savings algorithm of Clarke and Wright [4], which takes account of the time windows constraints. Note also that, at each iteration, all the columns with a negative reduced cost are incorporated into the master problem. A more sophisticated management of the columns might have reduced the computing time (see Desaulniers *et al.* [6]). Finally, note that we use a set covering model for both methods, so as to limit the size of the dual solution space.

The computational results allow us to draw several conclusions. The linear relaxation is tighter in the elementary path version 57 out of 87 times. The non-elementary path version is generally faster than the elementary path one, even if the CPU ratio is less than 2, 45 out of 77 times. In particular, our algorithm only failed to solve 10 instances out of the 87. The gap between both bounds is quite small, but the elementary path version is able to directly find 39 optimal solutions compared to only 16 for the non-elementary path version. Thus, it

	Non-elementary				Elementary					
Problem	♯ iter1	♯ col1	LB1	CPU1	♯ iter2	♯ col2	LB2	CPU2	Gap	Ratio
r101.25	10	122	*617.10	0.78	9	129	*617.10	0.67	0.00	0.86
r101.50	22	354	1043.37	3.29	22	472	1043.37	2.86	0.00	0.87
r101.100	43	1448	1631.15	8.63	34	1705	1631.15	13.2	0.00	1.53
r102.25	16	364	546.333	1.81	12	461	546.333	1.68	0.00	0.93
r102.50	25	1202	*909.00	4.29	19	1634	*909.00	6.35	0.00	1.48
r102.100	50	3828	*1466.60	28.67	32	8524	*1466.60	141.06	0.00	4.92
r103.25	23	603	454.067	2.91	20	701	*454.60	3.74	0.12	1.29
r103.50	32	1751	756.117	8.19	22	3856	769.233	24.69	1.71	3.01
r103.100	64	7432	1203.24	126.5	44	28706	1206.78	3073.52	0.29	24.30
r104.25	24	796	414.85	3.34	15	1232	*416.90	3.32	0.49	0.99
r104.50	44	3101	608.521	26.85	28	12988	619.077	338.84	1.71	12.62
r104.100	70	9915	937.22	424.01						
r105.25	15	228	*530.50	1.58	14	218	*530.50	1.56	0.00	0.99
r105.50	30	841	890.187	3.95	30	836	892.12	4.67	0.22	1.18
r105.100	45	2702	1341.19	15.06	44	3412	1346.14	38.73	0.37	2.57
r106.25	23	507	457.3	2.73	14	702	457.3	1.98	0.00	0.73
r106.50	35	1535	789.433	7.23	23	2544	791.367	11.79	0.24	1.63
r106.100	60	5893	1212.35	76.31	37	14234	1227.16	503.12	1.21	6.59
r107.25	23	822	415.125	3.13	18	762	*424.30	2.74	2.16	0.88
r107.50	39	2455	697.767	13.68	24	4724	707.26	37.2	1.34	2.72
r107.100	54	8573	1036.96	206.63						
r108.25	21	723	389.424	3.00	17	1496	396.821	4.89	1.86	1.63
r108.50	44	3495	578.482	34.31	26	23062	594.699	768.21	2.73	22.39
r108.100	64	11593	891.562	607.42						
r109.25	20	320	439.425	2.25	17	296	*441.30	1.93	0.42	0.86
r109.50	25	1363	727.515	4.69	28	1710	775.342	7.47	6.17	1.59
r109.100	42	4825	1097.49	37.13	35	7190	1134.11	114.53	3.23	3.08
r110.25	19	510	419.072	2.35	13	589	438.35	1.78	4.40	0.76
r110.50	28	1668	675.457	7.95	23	2447	695.061	14.03	2.82	1.76
r110.100	47	6033	1021.3	119.76	33	15006	1055.57	695.03	3.25	5.80
r111.25	19	580	412.815	2.56	16	551	427.283	2.53	3.39	0.99
r111.50	31	2283	658.752	10.75	26	4268	696.285	30.19	5.39	2.81
r111.100	55	8571	1005.93	168.13	41	32214	1034.73	2930.93	2.78	17.43
r112.25	19	674	365.03	2.78	18	1146	387.05	3.69	5.69	1.33
r112.50	39	2652	582.715	21.24	25	5712	614.851	69.55	5.23	3.27
r112.100	62	8879	892.508	469.98						

Table 5: Solution of the VRPTW by column generation for r-instances.

	Non-elementary				Elementary					
Problem	♯ iter1	♯ col1	LB1	CPU1	♯ iter2	♯ col2	LB2	CPU2	Gap	Ratio
c101.25	41	693	*191.3	4.77	41	605	*191.3	5.87	0.00	1.23
c101.50	66	1552	*362.4	10.55	52	1211	*362.4	9.25	0.00	0.88
c101.100	70	2498	*827.3	14.7	70	2557	*827.3	38.58	0.00	2.62
c102.25	39	1524	189.15	7.47	26	2304	*190.3	9.48	0.60	1.27
c102.50	48	2326	360.25	14.87	35	5587	*361.4	46.92	0.32	3.16
c102.100	100	6020	827.3	105.98						
c103.25	35	1425	187.857	8.99	24	6080	*190.3	67.28	1.28	7.48
c103.50	65	3844	360.25	36.81	40	11047	*361.4	507.51	0.32	13.79
c103.100	106	7037	826.3	298.82						
c104.25	34	1980	184.339	11.51	27	9195	*186.9	459.22	1.37	39.90
c104.50	50	4151	352.266	88.9						
c104.100	93	7738	821.53	616.73						
c105.25	35	644	*191.3	5.43	37	724	*191.3	5.11	0.00	0.94
c105.50	59	1600	*362.4	10.99	55	1368	*362.4	11.64	0.00	1.06
c105.100	106	3774	*827.3	35.21	97	3830	*827.3	90.91	0.00	2.58
c106.25	38	551	*191.3	4.36	49	781	*191.3	5.76	0.00	1.32
c106.50	45	1305	*362.4	6.79	55	1696	*362.4	10.63	0.00	1.57
c106.100	95	3956	*827.3	41.57	99	4582	*827.3	136.38	0.00	3.28
c107.25	43	824	*191.3	6.58	38	870	*191.3	4.7	0.00	0.71
c107.50	64	2107	*362.4	11.31	60	2035	*362.4	13.81	0.00	1.22
c107.100	103	4333	*827.3	43.92	111	5516	*827.3	155.1	0.00	3.53
c108.25	34	838	187.84	5.37	35	983	*191.3	5.58	1.81	1.04
c108.50	45	1827	359.81	10.33	49	1770	*362.4	16.79	0.71	1.63
c108.100	80	4144	817.352	53.45	102	6398	*827.3	250.17	1.20	4.68
c109.25	26	995	181.924	4.45	29	1280	*191.3	5.55	4.90	1.25
c109.50	49	2488	354.309	16.64	47	2730	*362.4	36.76	2.23	2.21
c109.100	81	5681	809.402	87.46	87	9866	*827.3	611.59	2.16	6.99

Table 6: Solution of the VRPTW by column generation for c-instances.

	Non-elementary				Elementary					
Problem	♯ iter1	♯ col1	LB1	CPU1	♯ iter2	♯ col2	LB2	CPU2	Gap	Ratio
rc101.25	26	344	390.15	3.58	25	402	406.625	3.58	4.05	1.00
rc101.50	34	714	826.613	4.8	25	892	850.021	4.02	2.75	0.84
rc101.100	46	2248	1567.45	12.82	38	2833	1584.09	22.83	1.05	1.78
rc102.25	31	618	347.067	4.27	26	628	*351.8	3.77	1.35	0.88
rc102.50	41	1225	706.606	10.55	32	1846	721.815	12.78	2.11	1.21
rc102.100	48	3773	1380.21	37.61	37	8574	1406.26	148.87	1.85	3.96
rc103.25	24	811	314.982	4.03	25	949	*332.8	7.09	5.35	1.76
rc103.50	45	1898	614.069	19.18	32	2810	645.281	40.7	4.84	2.12
rc103.100	63	5629	1170.32	141.42	33	21381	1225.54	1192.63	4.51	8.43
rc104.25	30	950	287.539	6.73	25	915	*306.6	8.08	6.22	1.20
rc104.50	52	2373	524.119	84.98	31	3771	*545.8	241.1	3.97	2.84
rc104.100	62	8381	1052.55	524.09						
rc105.25	22	535	408.525	2.72	18	508	*411.3	2.38	0.67	0.88
rc105.50	37	1189	746.314	6.94	28	1404	761.558	8.07	2.00	1.16
rc105.100	53	3524	1453.89	27.44	47	5124	1471.93	81.93	1.23	2.99
rc106.25	26	746	314.274	3.63	25	535	*345.5	3.73	9.04	1.03
rc106.50	35	1458	633.228	9.36	30	1368	664.433	8.77	4.70	0.94
rc106.100	48	3881	1248.96	39.91	41	5546	1318.8	109	5.30	2.73
rc107.25	34	846	281.289	7.68	31	687	*298.3	6.44	5.70	0.84
rc107.50	44	2139	570.665	29.06	36	2791	603.583	53.15	5.45	1.83
rc107.100	55	5475	1117.37	162.12	41	11568	1183.37	651.46	5.58	4.02
rc108.25	32	1025	270.573	10.43	22	1129	*294.5	9.47	8.12	0.91
rc108.50	45	2089	525.12	47.96	36	3308	541.167	240.31	2.97	5.01
rc108.100	57	7435	1035.71	350.82						

Table 7: Solution of the VRPTW by column generation for rc-instances.

is conceivable to expect the efficient solution of the VRPTW using a column generation methodology and our algorithm for the subproblem solution. The reader is referred to Kohl *et al.* [21] and Irnich [20] to compare our results with different kinds of bounds. Besides, the advantages of this algorithm, essentially the gap reduction and its adaptability for branching, could offset the loss in efficiency for many complex constrained vehicle routing problems.

Finally, let us mention that we have successfully experimented our algorithm for the solution of instances with up to 100 customers for two Vehicle Routing Problems with Profits, the so-called Selective Vehicle Routing Problem with Time Windows and Prize-collecting Vehicle Routing Problem with Time Windows (which respectively extend the Selective Traveling Salesman Problem, e.g., Laporte and Martello [22], and the Prize-collecting Traveling Salesman Problem, e.g., Balas [1]). See Gueguen [15] for a complete description of these applications.

6 Conclusion

In this paper, we have presented an algorithm for the optimal solution of the Elementary Shortest Path Problem with Resource Constraints (ESPPRC). Besides the obvious interest in solving constrained elementary shortest paths problems, this algorithm aims to be used within column generation schemes, for the solution of important problems such as vehicle routing. In this context, it has two main advantages. Firstly, it reduces the duality gap, compared to the usual approaches that are based on the SPPRC bound. Secondly, it enables the use of a column generation solution methodology for some special problems or with some special branching schemes that cannot rely on SPPRC solutions.

The algorithm is based on Desrochers' label correcting algorithm [7] and new resources are introduced to enforce the elementary path constraint, as proposed by Beasley and Christofides [2]. We have strongly improved the efficiency of the algorithm with the introduction of the idea of unreachable nodes.

Our computational study performed on a variety of data demonstrates the efficiency of our algorithm and shows that our approach can successfully be used for solving ESPPRC instances of moderate sizes. We also have obtained satisfactory results when applying our algorithm to the solution of the Vehicle Routing Problem with Time Windows. Embedded in a column generation procedure, our algorithm significantly reduces the gap with a reasonable computation time, when used instead of a SPPRC solution algorithm. Finally, our approach has also proved its efficiency concerning the solution of various problems, such as the VRP with Profits, that we have not detailed in this paper.

Acknowledgments

This work was started while Pierre Dejax, Dominique Feillet and Cyrille Gueguen were members of the Laboratoire Productique Logistique at Ecole Centrale Paris. The authors wish to thank Maciej Witucki for numerous suggestions when applying our algorithm to his own problem and Moshe Dror for valuable comments on the paper. The authors also thank two anonymous referees for their numerous and very helpful comments and suggestions.

References

[1] Balas E. The Prize Collecting Traveling Salesman Problem. II: Polyhedral results. *Networks*, 25(4):199–216, 1995.

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

[3] Borndörfer R., Grötschel M., and Löbel A. Scheduling duties by adaptative column generation. Technical Report ZIB-report 01-02, Konrad-Zuze-Zentrum für Informationstechnik, Berlin, 2001.

[4] Clarke R.M. and Wright J.W. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12:568–581, 1964.

[5] Current J. and Marsh M. Multiobjective transportation network design and routing problems: Taxonomy and annotation. *European Journal of Operational Research*, 65:4–19, 1993.

[6] Desaulniers G., Desrosiers J., and Solomon M.M. Accelerating strategies in column generation methods for vehicle routing and crew scheduling problems. Technical Report G-99-36, GERAD, 1999.

[7] Desrochers M. An algorithm for the shortest path problem with resource constraints. Technical Report G-88-27, GERAD, 1988.

[8] Desrochers M., Desrosiers J., and Solomon M. A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research*, 40(2):342–354, 1992.

[9] Dror M. Note on the complexity of the shortest path models for column generation in vrptw. *Operations Research*, 42(5):977–978, 1994.

[10] Eurodécision. *LPToolKit, User's Guide and Technical Reference*, 3.6 edition, 1998.

[11] Feillet D., Dejax P., and Gendreau M. Traveling salesman problems with profits: An overview. Technical Report CER 00-04A, Laboratoire Productique Logistique, Ecole Centrale Paris, 2000.

[12] Gelinas S., Desrochers M., Desrosiers J., and Solomon M.M. A new branching strategy for time constrained routing problems with application to backhauling. *Annals of Operations Research*, 61:91–109, 1995.

[13] Gondran M. and Minoux M. *Graphes et Algorithmes*. Eyrolles, 1985.

[14] Graves G.W., McBride R.D., Gershkoff I., Anderson D., and Mahidhara D. Flight crew scheduling. *Management Science*, 39(6):657–682, 1993.

[15] Gueguen C. *Méthodes de résolution exacte pour les problèmes de tournées de véhicules*. PhD thesis, Laboratoire Productique Logistique, Ecole Centrale Paris, 1999.

[16] Handler G.Y. and Zang I. A dual algorithm for the constrained shortest path problem. *Networks*, 10:293–310, 1980.

[17] Hansen P., Jaumard B., and Vovor T. Solving the bicriterion shortest path problem from both ends. Technical Report G-98-17, GERAD, 1998.

[18] ILOG Inc., CPLEX Division, Incline Village. *ILOG, Using the CPLEX Callable Library*, 6.0 edition, 1998.

[19] Irnich S. The shortest path problem with k-cycle elimination (k≥3): Improving a branch and price algorithm for the VRPTW. In *TRISTAN IV*, volume 3, pages 571–574, 2001.

[20] Irnich S. The shortest path problem with k-cycle elimination (k≥3): Improving a branch and price algorithm for the VRPTW. Technical report, Lehr- und Forschungsgebiet Unternehmensforschung, Aachen university of Technology, 2001.

[21] Kohl N., Desrosiers J., Madsen O.B.G., Solomon M.M., and Soumis F. 2-path cuts for the vehicle routing problem with time windows. *Transportation Science*, 33(1):101–116, 1999.

[22] Laporte G. and Martello S. The Selective Travelling Salesman Problem. *Discrete Applied Mathematics*, 26:193–207, 1990.

[23] Lavoie S., Minoux M., and Odier E. A new approach for crew pairing problems by column generation with an application to air transportation. *European Journal of Operational Research*, 35:45–58, 1988.

[24] Solomon M.M. *Vehicle Routing and Scheduling with Time Window Constraints: Models and Algorithms.* PhD thesis, Department of Decision Sciences, University of Pennsylvania, 1983.

[25] Warburton A. Approximation of pareto optima in multiple-objective, shortest-path problems. *Operations Research*, 35(1):70–79, 1987.

[26] Ziegelmann M. *Constrained Shortest Paths and Related Problems.* PhD thesis, Universität des Saarlandes, 2001.