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

Dominique Feillet
Laboratoire d'Informatique d'Avignon, Université d'Avignon, 339 Chemin des Meinajariés, BP 1228, 84911 Avignon Cedex 9, France

Pierre Dejax
Département automatique et Productique Ecole des mines de Nantes, 4, rue Alfred Kastler, BP20722, 44307 Nantes Cedex 3, France

Michel Gendreau
Centre de recherche sur les transports, Université de Montréal, C.P. 6128 succursale Centre-ville, Montréal, Canada H3C 3J7

Cyrille Gueguen
Département de recherche opérationnelle, Direction Générale des Systèmes d'Information, AIR FRANCE, 1, av. du Maréchal Devaux, 91551 Paray-Vieille-Poste Cedex, France

In this article, 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 (nonelementary) 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. © **2004 Wiley Periodicals, Inc. NETWORKS, Vol. 44(3), 216–229 2004**

Keywords: shortest path; column generation; vehicle routing

1. INTRODUCTION

In this article, 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. [10] for the Vehicle Routing Problem with Time Windows (VRPTW), by Lavoie et al. [26] for the airline crew pairing, by Graves et al. [16] 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 pseudopolynomial 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 [9] 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

Received April 2000; accepted June 2004
Correspondence to: D. Feillet; e-mail: dominique.feillet@lia.univ-avignon.fr
DOI 10.1002/net.20033
Published online in Wiley InterScience (www.interscience.wiley.com).

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 Lagrangian relaxation technique. This approach takes advantage of the effectiveness of algorithms that solve the unconstrained shortest path problem. Handler and Zang [18] and Beasley and Christofides [2] address the problem this way, as Borndörfer et al. [3], Dumitrescu and Boland [12], or Ziegelmann [30] more recently. Note that Ziegelmann [30] 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 multiobjective shortest path problems, which have received the attention of many researchers. Indeed, the aim in these problems is also to generate nondominated paths (i.e., Pareto optimal paths). However, the methods developed in the literature usually concern multiobjective shortest path problems on graphs with nonnegative lengths (see, i.e., Warburton [29] and Hansen et al. [19]). The interested reader may find an exhaustive review of multiobjective shortest path problems in Current and Marsh [7].

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 [11] proved that the ESPPRC is NP-hard in the strong sense. To our knowledge, no article has been dedicated to its solution when the graph contains negative costs, which frequently happens in the context of column generation. Gondran and Minoux [15] did, however, propose a suboptimal approach for solving the Elementary Shortest Path Problem (without resource constraint) in graphs that might contain negative cost cycles.

The algorithm proposed in this article is able to solve optimally the ESPPRC in general graphs, even if they contain negative cost cycles. It is adapted from Desrochers' SPPRC label correcting algorithm [9]. In Section 2, we describe formally the ESPPRC. Section 3 then presents our solution algorithm. The possibility of using our algorithm in the context of column generation is then illustrated in Section 4, with a focus on the VRPTW case. Computational results of Section 5 conclude the article, with experiments conducted on pure ESPPRC instances and on VRPTW instances.

2. 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. The ESPPRC can be described with the following model:

$$\text{minimize} \sum_{(v_i,v_j)\in A} c_{ij}x_{ij} \tag{1}$$

subject to

$$\sum_{(v_i,v_j)\in A} x_{ij} - \sum_{(v_j,v_i)\in A} x_{ji} = 0 \quad \forall\, v_i \in V\backslash\{p, d\}, \tag{2}$$

$$\sum_{(p,v_j)\in A} x_{pj} = 1, \tag{3}$$

$$\sum_{(v_j,d)\in A} x_{jd} = 1, \tag{4}$$

$$t_i^l + d_{ij}^l - t_j^l + Mx_{ij} \leq M \quad \forall\, l \in \{1, \ldots, L\},\ (v_i, v_j) \in A, \tag{5}$$

$$a_i^l \leq t_i^l \leq b_i^l \quad \forall\, l \in \{1, \ldots, L\},\ v_i \in V, \tag{6}$$

$$x_{ij} \in \{0, 1\} \quad \forall\, (v_i, v_j) \in A, \tag{7}$$

where x_{ij} variables represent flow in the network and t_i^l variables stand for resource consumption, and M is a big number.

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

3. A LABEL CORRECTING ALGORITHM TO SOLVE THE ESPPRC

3.1. Notation and Principle of Desrochers' Algorithm for the SPPRC

Desrochers' algorithm [9] is a label correcting reaching algorithm. It is an extension of the Ford-Bellman algorithm, taking resource constraints into account. Each node receives labels throughout the algorithm, that stand for partial paths and indicate the cost and the resource consumption of these paths. Nodes are iteratively treated until no new labels are created. When a node is treated, all its new labels are extended toward every possible successor node. Dominance

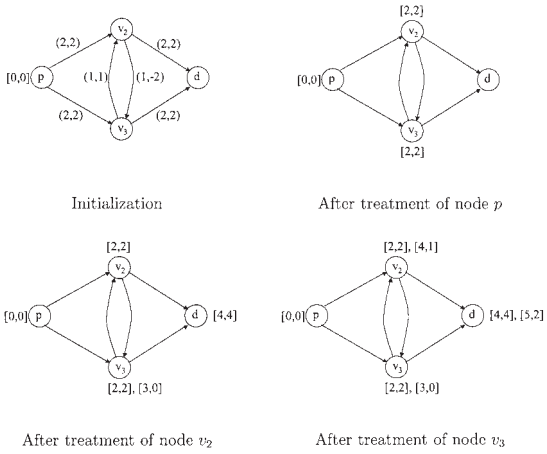

FIG. 1. First iterations of Desrochers' algorithm.

rules are introduced to limit the proliferation of labels. We need some notation for the description of these rules.

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 $C_i = C(T_i^1, T_i^2, \ldots, T_i^L)$. The dominance rule works as follow. 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 $C'_i \leq C^*_i$, $T'^l_i \leq T^{*l}_i$ 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 nondominated labels, that is, nondominated paths. Figure 1 illustrates the first iterations of this algorithm on a simple graph with four nodes ($p = v_1$, v_2, v_3, $d = v_4$), and a single resource.

In this figure, the first picture indicates arc resource consumptions and costs in brackets. Labels are represented in square brackets, with their resource consumption level and their cost.

3.2. Adaptation to the ESPPRC Context

Let us first say that adjusting Desrochers' label correcting algorithm [9] 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 pseudopolynomial 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 to labels an extra binary resource for each node $v_k \in V$: this resource would take value 0 initially and would be set to 1 when the node is visited in the label. 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 article, we adapt this idea to Desrochers' label correcting algorithm [9] 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), because our purpose was to focus on the elementary path question.

We change the label definition and add n extra binary resources, one for each node $v_k \in V$. 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 [9]. Note that $s_i = \sum_{k=1}^n V_i^k$, and that this resource is only needed for computational purposes, because 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 nondominated paths are the extension of a nondominated 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 nondominated 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. Figure 2 illustrates the new definition of labels. The four intermediate digits in labels indicate node visit (resource s_i is not mentioned for sake of simplicity).

3.3. Improved Definition of the Labels

In this section, we once more propose another definition of the labels, to limit their number throughout the process.

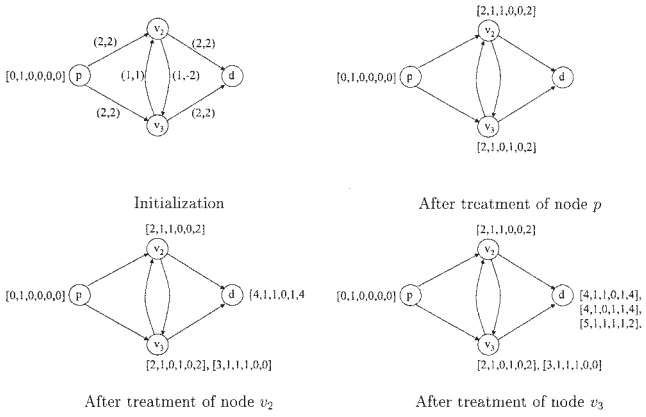

FIG. 2. Illustration of the algorithm with the elementary condition.

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 that 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 3.2, 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 formalize 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 because 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 nondominated paths.

Claim 1. *During the execution of the modified algorithm, we need only to consider nondominated 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 $l = 1, \ldots, L$. Because $T_j'^l = \max\{a_j^l, T_i'^l + d_{ij}^l\}$ 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 $V_i^j = 0$, because (R_i', C_i') dominates (R_i, C_i) and because 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:

1. 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 $V_j^k = 1$.

2. Second, if there exists $l \in \{1, \ldots, L\}$ with $T'^{l}_{j} + d^{l}_{jk} > b^{l}_{k}$, $T^{l}_{j} + d^{l}_{jk} > b^{l}_{k}$, because $T'^{l}_{j} \leq T^{l}_{j}$ 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 $V'^{k}_{j} \leq V^{k}_{j}$ 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). ∎

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, 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.

3.4. 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 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 nondominated labels in the list of labels Λ.

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

```
ESPPRC(p)
1  INITIALIZATION
2  Λ_p ← {(0, ..., 0)}
3  for all v_i ∈ V \ {p}
4     do Λ_i ← ∅
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.

4. SHORTCOMINGS OF THE CLASSICAL COLUMN GENERATION SOLUTION APPROACH FOR SOME VEHICLE ROUTING PROBLEMS

Many of the most efficient solution algorithms for vehicle routing problems are based on a decomposition scheme and rely on the solution of SPPRC instances as subproblems. A representative (and seminal) work on this topic is from Desrochers et al. [10]. This work addresses the solution of the VRPTW with a column generation scheme where new columns are found solving SPPRC instances. Before mentioning some other examples of similar schemes and explaining the part that could play the ESPPRC in these schemes, we need to recall some issues of Desrochers et al. [10] approach.

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. Desrochers et al. [10] first formulate the VRPTW using the Classical Set Partitioning Model:

$$\text{minimize} \sum_{r_k \in \Omega} c_k x_k \tag{8}$$

subject to

$$\sum_{r_k \in \Omega} a_{ik} x_k = 1 \quad \forall\, v_i \in V, \tag{9}$$

$$x_k \in \{0, 1\} \quad \forall\, r_k \in \Omega, \tag{10}$$

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. The size of Ω implies the use of a column generation procedure for computing the linear relaxation of this model. The decomposition scheme, with the definition of Ω, then results in formulating the subproblem as an ESPPRC.

TABLE 1. Solution of the ESPPRC for r-instances with 50 customers.

Problem	No. of arcs	Density	% neg arcs	CPU_{BI}	CPU	No. of labels
r101.50	813	31.88	7.99	0.01	0	10
r102.50	1504	58.98	8.64	0.48	0.03	21
r103.50	1970	77.25	7.56	15.01	0.46	43
r104.50	2398	94.03	5.83	52.37	3.26	32
r105.50	1068	41.88	7.49	0.01	0	13
r106.50	1696	66.5	6.66	0.21	0.06	23
r107.50	2092	82.03	7.12	43.25	1.51	65
r108.50	2437	95.56	6.11		45.76	53
r109.50	1534	60.15	6.51	0.04	0.03	18
r110.50	2019	79.17	7.92	3.82	0.14	57
r111.50	2035	79.8	7.32	9.48	0.18	46
r112.50	2528	99.13	4.86	5.51	0.2	26
r201.50	1519	59.56	5.46	5.46	0.01	14
r202.50	1985	77.84	7.9		60.26	107
r203.50	2296	90.03	5.13		59.42	90
r205.50	1879	73.68	7.5		0.67	69
r206.50	2184	85.64	5.76		6.09	97
r209.50	2151	84.35	8.46		26.87	66
r210.50	2189	85.84	5.52		21.28	57

Instead of the set partitioning model described above, Desrochers et al. [10] introduce a set covering formulation, where they replace constraints (9) with constraints

$$\sum_{r_k \in \Omega} a_{ik} x_k \geq 1 \quad \forall\, v_i \in V \tag{11}$$

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. [10] 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. [14]. The computational results of Section 5.2 will permit us to assess this weakening of the lower bound for the VRPTW.

Apart from Desrochers et al.'s article [10], many decomposition procedures have been proposed in the literature for solving the VRPTW or other vehicle routing problems, with similar schemes. Most of them rely on column generation, while some have investigated other types of decomposition as Lagrangian relaxation (Kallehauge et al. [22]). One can cite Cordeau et al. [6] or Lübbecke and Desrosiers [27] for many references on the subject.

In all these cases, the efficiency of the solution algorithm is penalized by the weakening of the lower bound resulting from the nonelementary modeling. In the context of the VRPTW, some work has been carried out to limit this weakening, with the help of valid inequalities (Kohl et al. [23]) or by eliminating cycles that contain no more than two arcs (e.g., Desrochers et al. [10]) or no more than k arcs (Irnich [20]). All of these attempts are successful, but having an efficient ESPPRC solution algorithm would be still more satisfactory.

It is all the more true because the lower bound weakening

TABLE 2. Solution of the ESPPRC for c-instances with 50 customers.

Problem	No. of arcs	Density	% neg arcs	CPU_{BI}	CPU	No. of labels
c101.50	1218	47.76	15.1	61.73	0.01	50
c102.50	1749	68.58	15.2		24.29	169
c105.50	1334	52.31	15.66		0.01	43
c106.50	1273	49.92	14.37	182.82	0.03	45
c107.50	1444	56.62	13.15		0.03	50
c108.50	1643	64.43	12.41		0.09	46
c109.50	1896	74.35	12.5		0.2	67
c201.50	1340	52.54	9.4		0.2	352
c202.50	1891	74.15	10.1		254.48	284
c205.50	1477	57.92	15.5		1.1	422
c206.50	1590	62.35	11.13		0.73	246

TABLE 3. Solution of the ESPPRC for r-instances with 100 customers.

Problem	No. of arcs	Density	% neg arcs	CPU_{BI}	CPU	No. of labels
r101.100	3244	32.11	10.32	3.51	0.06	28
r102.100	5813	57.55	10.23		236.78	272
r105.100	4261	42.18	10.79		0.14	49
r106.100	6559	64.94	8.69		167.92	142
r109.100	6033	59.73	11.02		2.07	295
r110.100	7966	78.87	8.57		66.5	350
r201.100	5918	58.59	8.76		11.15	762

would be even more detrimental for some other vehicle routing problems. An illustrative example is the Vehicle Routing Problems with Profits class (see Feillet et al. [13]), 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. Furthermore, 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, some optimal elementary routes might always be dominated by nonelementary routes, as soon as the branching scheme does not prohibit these nonelementary routes. These routes might then never enter the pool of columns. 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. Some other examples could certainly be mentioned.

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 [28]. 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. Each family of instances is divided into two parts, the first part (r101–r112, c101–c109, rc101–rc108) having smaller time windows than the second part (r201–r211, c201–c208, rc201–rc208). 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 nontruncated values would not have affected our computation times. Two resources are defined: time and load. Time consumption d_{ij}^1 is given by $d_{ij}^1 = st_i + dist_{ij}$ where v_i and v_j are two customers and st_i represents the service time for customer v_i. Load consumption d_{ij}^2 is given by $d_{ij}^2 = q_j$ where q_j is the quantity to deliver to customer v_j. Each customer v_i receives a time interval $[a_i^1, b_i^1]$ 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

TABLE 4. Solution of the ESPPRC for c-instances with 100 customers.

Problem	No. of arcs	Density	% neg arcs	CPU_{BI}	CPU	No. of labels
c101.100	4516	44.71	8.37		0.12	70
c102.100	6684	66.17	6.98		57.98	132
c105.100	5052	50.01	6.92		0.18	97
c106.100	5423	53.69	7.56		0.34	128
c107.100	5619	55.63	6.47		0.25	67
c108.100	6343	62.8	6.71		0.56	64
c109.100	7455	73.81	7.37		2.37	121
c201.100	5222	51.7	5.82		1.14	445
c205.100	5699	56.42	6.98		5.51	400
c206.100	6213	61.51	8.49		25.87	384

TABLE 5. Solution of the VRPTW by column generation for r1-instances.

	Nonelementary				Elementary					
Problem	LB	CPU	No. of iter	No. of col	LB	CPU	No. of iter	No. of col	Gap	Ratio
r101-025	**617.1**	0.156	12	95	**617.1**	0.047	11	98	0	0.301
r101-050	1043.37	0.359	18	358	1043.37	0.421	19	403	0	1.17
r101-100	1631.15	3.719	39	1519	1631.15	6.578	33	1736	0	1.77
r102-025	546.333	0.188	15	270	546.333	0.188	14	391	0	1
r102-050	**909**	1.688	27	897	**909**	1.609	17	1238	0	0.953
r102-100	**1466.6**	18.672	57	2994	**1466.6**	49.563	40	5419	0	2.65
r103-025	454.067	0.562	21	595	**454.6**	0.515	20	710	0.117	0.916
r103-050	756.117	4.921	38	1577	769.233	9.89	25	2734	1.71	2.01
r103-100	1203.24	64.266	79	4571	1206.78	386.89	53	11800	0.294	6.02
r104-025	414.85	0.953	23	628	**416.9**	1.125	16	894	0.492	1.18
r104-050	608.521	17.687	51	2476	619.077	232.219	42	7222	1.71	13.1
r104-100	937.31	379.468	116	8774						
r105-025	**530.5**	0.14	16	228	**530.5**	0.109	17	214	0	0.779
r105-050	890.187	1	28	823	892.12	1.219	27	962	0.217	1.22
r105-100	1341.19	13.265	48	2899	1346.14	23.375	42	3200	0.368	1.76
r106-025	457.3	0.375	17	454	457.3	0.36	16	559	0	0.96
r106-050	789.433	3.156	31	1241	791.367	4.891	24	1993	0.244	1.55
r106-100	1212.27	58.375	68	4510	1226.91	199.016	45	8360	1.19	3.41
r107-025	415.125	0.672	19	622	**424.3**	0.828	22	702	2.16	1.23
r107-050	697.767	7.406	38	1846	707.26	15.984	27	2796	1.34	2.16
r107-100	1036.96	164.64	84	5680	1053.45	3560.59	60	14198	1.57	21.6
r108-025	389.424	1.282	24	761	396.821	1.735	18	1139	1.86	1.35
r108-050	578.482	23.938	49	2884	594.699	338.297	33	6453	2.73	14.1
r108-100	891.562	635.578	129	9630						
r109-025	439.425	0.328	21	347	**441.3**	0.171	16	360	0.425	0.521
r109-050	727.515	3.734	33	1496	775.342	3.5	26	1511	6.17	0.937
r109-100	1097.41	49.719	66	4568	1134.23	91.047	47	6460	3.25	1.83
r110-025	419.072	0.719	21	612	438.35	0.688	17	676	4.4	0.957
r110-050	675.457	6.219	36	1614	695.061	8.703	25	2444	2.82	1.4
r110-100	1021.3	131.875	81	5226	1055.57	482.047	52	9878	3.25	3.66
r111-025	412.815	0.703	24	602	427.283	0.657	23	613	3.39	0.935
r111-050	658.752	9	39	2006	696.285	17.844	32	2954	5.39	1.98
r111-100	1005.93	162.438	96	6607	1034.73	974.937	69	14077	2.78	6
r112-025	365.03	1.797	28	871	387.05	2.203	20	1134	5.69	1.23
r112-050	582.715	20.532	51	2189	614.851	58.062	32	4110	5.23	2.83
r112-100	892.545	510.063	115	8963						

with a 1.6-GHz processor and with 256 Megabytes of RAM. Algorithms were implemented in C++ with Visual C++ 6.0 Compiler. Linear programs are solved with CPLEX 7.5. 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 [28] 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 paths starting from the depot, coming back to an artificial copy of the depot and satisfying resource constraints. All r- and c-instances are addressed, with 50 and 100 customers.

Results are summarized in Tables 1 to 4. For each instance, we first give the number of arcs in the graph, its density and the percentage of arcs with a negative cost. The computation times (in seconds) needed for the solution are respectively given in columns CPU_{BI} and CPU for the version of the algorithm before improvement (section 3.2) and for the final version of the algorithm. Column *No. of labels* gives the cardinality of the solution set (containing all the Pareto optimal paths), that is, the number of labels on the destination node at the end of the algorithm. Missing instances or empty cells indicate that the problem was not solved in the imparted time.

Our algorithm succeeds in solving 30 instances out of 40 for instances with 50 customers, while the version without the improvement described in Section 3.3 permits only to solve 14 instances. When 100 customers are considered, we solve 17 instances out of 40, against 1 without the improvement. Furthermore, these tables highlight the significant reduction of computing times resulting from the improvement.

TABLE 6. Solution of the VRPTW by column generation for c1-instances.

	Nonelementary				Elementary					
Problem	LB	CPU	No. of iter	No. of col	LB	CPU	No. of iter	No. of col	Gap	Ratio
c101-025	**191.3**	0.703	42	647	**191.3**	0.531	42	647	0	0.755
c102-050	**362.4**	2.328	62	1301	**362.4**	2.844	63	1337	0	1.22
c101-100	**827.3**	13.938	121	2723	**827.3**	18.703	86	2652	0	1.34
c102-25	189.15	2.297	32	1188	**190.3**	2.766	22	1677	0.604	1.2
c102-50	360.3	5.781	56	1887	**361.4**	87.688	57	6980	0.318	4.53
c102-100	**827.3**	78.703	109	5174	**827.3**	1066.25	85	12315	0	13.5
c103-025	187.857	4.625	35	1452	**190.3**	24.235	30	3598	1.28	5.24
c103-050	360.25	19.36	73	2645	**361.4**	87.688	57	6980	0.318	4.53
c103-100	**826.3**	268.578	164	8147						
c104-025	184.339	7.625	36	1599	**186.9**	127.453	31	5156	1.37	16.7
c104-050	352.266	90.219	83	4197						
c104-100	821.53	655.61	189	10516						
c105-025	**191.3**	0.703	32	676	**191.3**	0.656	37	880	0	0.933
c105-050	**362.4**	2.203	64	1224	**362.4**	4.515	55	1487	0	2.05
c105-100	**827.3**	15.516	87	3267	**827.3**	40.39	102	3828	0	2.6
c106-025	**191.3**	0.437	32	544	**191.3**	0.562	38	663	0	1.29
c106-050	**362.4**	2.703	62	1376	**362.4**	2.656	60	1446	0	0.983
c106-100	**827.3**	27.61	103	3814	**827.3**	80.359	116	5352	0	2.91
c107-025	**191.3**	1.094	42	846	**191.3**	1.391	34	932	0	1.27
c107-050	**362.4**	4.516	76	1421	**362.4**	4.797	56	1737	0	1.06
c107-100	**827.3**	25.406	98	4274	**827.3**	39.89	87	4055	0	1.57
c108-025	187.84	1.969	32	908	**191.3**	1.5	31	883	1.81	0.762
c108-050	359.81	9.469	71	2404	**362.4**	9.843	58	1895	0.715	1.04
c108-100	817.352	59.515	105	4698	**827.3**	127.828	99	5585	1.2	2.15
c109-025	181.924	3.094	31	1190	**191.3**	3.391	36	1396	4.9	1.1
c109-050	354.309	17.281	62	2925	**362.4**	26.953	55	3443	2.23	1.56
c109-100	809.402	104.125	117	5975	**827.3**	446.047	115	10508	2.16	4.28

TABLE 7. Solution of the VRPTW by column generation for rc1-instances.

	Nonelementary				Elementary					
Problem	LB	CPU	No. of iter	No. of col	LB	CPU	No. of iter	No. of col	Gap	Ratio
rc101-025	390.15	0.515	30	388	406.625	0.312	20	341	4.05	0.606
rc101-050	826.613	1.562	33	766	850.021	1.156	26	900	2.75	0.74
rc101-100	1567.45	11.235	50	2446	1584.09	16.891	42	2829	1.05	1.5
rc102-025	347.067	1.172	31	507	**351.8**	0.89	23	674	1.35	0.759
rc102-050	707.003	6.813	40	1176	721.815	5.812	31	1746	2.05	0.853
rc102-100	1380.23	37.157	59	3346	1406.26	87.672	45	5633	1.85	2.36
rc103-025	313.976	2.391	26	707	**332.8**	1.094	19	756	5.66	0.458
rc103-050	613.259	21.657	51	1728	645.281	25.907	29	2195	4.96	1.2
rc103-100	1170.32	153.844	90	5345	1225.65	800.922	57	10217	4.51	5.21
rc104-025	291.676	3.282	31	775	**306.6**	2.89	22	829	4.87	0.881
rc104-050	524.119	82.547	66	2663	**545.8**	128.079	42	3378	3.97	1.55
rc104-100	1052.55	661.266	121	7404						
rc105-025	408.525	0.781	19	490	**411.3**	0.546	19	414	0.675	0.699
rc105-050	746.314	4.281	35	1144	761.558	3.703	22	1331	2	0.865
rc105-100	1453.89	25.891	59	2924	1471.92	51.125	44	4809	1.22	1.97
rc106-025	314.274	1.594	27	698	**345.5**	0.75	23	546	9.04	0.471
rc106-050	633.228	9.39	42	1453	664.433	5.672	31	1671	4.7	0.604
rc106-100	1248.96	46.891	67	3574	1318.8	82.953	50	5172	5.3	1.77
rc107-025	281.289	5.688	36	971	**298.3**	2.484	30	864	5.7	0.437
rc107-050	570.665	35.984	58	2101	603.583	25.359	35	2338	5.45	0.705
rc107-100	1117.37	173.39	86	4935	1183.37	535.735	58	8491	5.58	3.09
rc108-025	270.573	8.25	33	1053	**294.5**	4.719	22	1269	8.12	0.572
rc108-050	525.12	59.625	59	2654	541.167	252.469	42	3456	2.97	4.23
rc108-100	1035.71	445.281	106	6114						

TABLE 8. Solution of the VRPTW by column generation for r2-instances.

	Nonelementary				Elementary					
Problem	LB	CPU	No. of iter	No. of col	LB	CPU	No. of iter	No. of col	Gap	Ratio
r201-025	448.5	2.235	20	1414	460.1	0.703	25	569	2.52	0.315
r201-050	754.098	8.187	35	2630	**791.9**	9	33	2639	4.77	1.1
r201-100	1080.6	130.735	81	9620	1140.3	205.906	75	10101	5.24	1.57
r202-025	374.092	4.469	26	2021	**410.5**	2.297	18	1384	8.87	0.514
r202-050	638.62	29.891	50	4536	**698.5**	70.641	35	7747	8.57	2.36
r202-100	935.278	1174.83	133	14540						
r203-025	337.51	8.844	34	2253	**391.4**	5.516	15	1985	13.8	0.624
r203-050	538.526	96.687	70	5779	598.583	538.344	35	14984	10	5.57
r203-100										
r204-025	303.99	49.562	46	4733	350.475	89.875	23	5941	13.3	1.81
r204-050										
r204-100										
r205-025	365.475	7.437	29	2488	390.6	2.843	28	1332	6.43	0.382
r205-050	595.548	34.875	51	5136	682.85	38.032	32	5759	12.8	1.09
r205-100	838.596	1116.81	149	19734	939.124	2762.55	106	27744	10.7	2.47
r206-025	319	30.704	36	4557	373.6	9.734	21	2808	14.6	0.317
r206-050	532.326	132.047	71	7970	626.343	255.032	33	11653	15	1.93
r206-100										
r207-025	309.609	51.828	46	4153	360.05	24.578	25	3893	14	0.474
r207-050	467.233	500.5	106	10922						
r207-100										
r208-025	291.195	1062.89	84	12172	**328.2**	310.734	34	8237	11.3	0.292
r208-050										
r208-100										
r209-025	327.253	16.656	31	3454	364.05	7.735	23	2514	10.1	0.464
r209-050	535.216	127.328	73	7631	599.825	174.672	43	8066	10.8	1.37
r209-100										
r210-025	340.505	6.343	29	2541	404.175	3.547	20	1875	15.8	0.559
r210-050	531.817	88.703	62	6995	636.1	263.718	29	10771	16.4	2.97
r210-100	753.697	3392.58	193	23423						
r211-025	299.447	150.906	54	6092	341.327	41.375	25	4653	12.3	0.274
r211-050	457.355	405.406	81	9324						
r211-100										

Finally, these results also 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 [28], embedded in a column generation procedure devoted to the solution of the VRPTW. Because our objective is just to test the shortest path algorithm, that is, 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 written in bold in the result tables.

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 4). In both methods, 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. At each iteration, the subproblem is stopped prematurely when 500 labels have been extended to the depot with a negative cost. All nondominated columns are then incorporated into the Master Problem. Finally, we use a set covering model, so as to limit the size of the dual solution space.

Let us recall that our main objective here is to evaluate the possibility of enforcing the elementary condition when solving a vehicle routing problem with a column generation approach. We do not expect any superiority of our algorithm compared to the most efficient algorithms proposed in the literature (see, e.g., Kohl et al. [23], Larsen [25], Cook and Rich [5], Kallehauge et al. [22]). Actually, devising an efficient algorithm requires far more than a standard implementation of the method, as illustrated in Desaulniers et al. [8].

TABLE 9. Solution of the VRPTW by column generation for c2-instances.

	Nonelementary				Elementary					
Problem	LB	CPU	No. of iter	No. of col	LB	CPU	No. of iter	No. of col	Gap	Ratio
c201-025	**214.7**	1.968	57	1326	**214.7**	1.891	57	1326	0	0.961
c201-050	**360.2**	134.938	165	9323	**360.2**	130.859	173	9443	0	0.97
c201-100	**589.1**	1414.83	458	30895	**589.1**	1121.22	306	24492	0	0.792
c202-025	214.7	8.328	64	2658	**214.7**	18.828	45	3708	0	2.26
c202-050	360.2	134.813	133	9236						
c202-100	**589.1**	3164.28	386	40715						
c203-025	213.775	41.407	64	3492	**214.7**	117.781	36	6439	0.431	2.84
c203-050	359.8	923.078	242	15324						
c203-100										
c204-025	207.156	101.968	69	4363						
c204-050										
c204-100										
c205-025	196.525	10.766	58	2856	**214.7**	12.609	76	3266	8.47	1.17
c205-050	341.763	506.594	153	16088	**359.8**	758.672	165	20002	5.01	1.5
c205-100	582.363	2761.05	311	35998						
c206-025	194.015	24.078	59	4002	**214.7**	14.234	74	3477	9.63	0.591
c206-050	338.361	616.469	152	16851	**359.8**	1689.81	185	30285	5.96	2.74
c206-100										
c207-025	200.442	60.156	61	5547	**214.5**	64.875	53	6292	6.55	1.08
c207-050	348.625	920.469	149	17356						
c207-100										
c208-025	183.817	51.61	63	5208	**214.5**	29.843	51	5002	14.3	0.578
c208-050	331.049	551.875	147	15484						
c208-100										

TABLE 10. Solution of the VRPTW by column generation for rc2-instances.

	Nonelementary				Elementary					
Problem	LB	CPU	No. of iter	No. of col	LB	CPU	No. of iter	No. of col	Gap	Ratio
rc201-025	316.803	1.719	26	1280	**360.2**	0.562	21	770	12	0.327
rc201-050	536.631	16.765	57	3254	**684.8**	8.312	51	2584	21.6	0.496
rc201-100	1108.69	137.203	102	9134	1255.94	282.093	119	10031	11.7	2.06
rc202-025	256.977	155.859	42	3452	**338**	3.953	22	1791	24	0.0254
rc202-050	433.243	328.953	90	8386	**613.6**	46.796	36	6245	29.4	0.142
rc202-100	888.001	1613.5	161	18735	1088.08	2411.38	106	31972	18.4	1.49
rc203-025	180.179	506.391	66	6965	**326.9**	63.125	23	4547	44.9	0.125
rc203-050	343.93	3201.86	148	16567						
rc203-100										
rc204-025	159.942	1348.83	91	10021	**299.7**	121.406	25	6728	46.6	0.09
rc204-050										
rc204-100										
rc205-025	268.48	6.875	29	2176	**338**	1.453	23	1045	20.6	0.211
rc205-050	484.314	98.89	66	7231	**630.2**	28.078	38	4749	23.1	0.284
rc205-100	967.469	712.938	144	15144	1147.61	1492.28	107	20099	15.7	2.09
rc206-025	189.548	69.36	72	5897	**324**	1.484	25	1065	41.5	0.0214
rc206-050	363.453	1134.86	143	15268	**610**	31.079	40	5272	40.4	0.0274
rc206-100	852.817	1691.28	192	23838						
rc207-025	173.319	169.765	60	7643	**298.3**	5.531	23	1474	41.9	0.0326
rc207-050					**558.6**	471.469	59	16961		
rc207-100										
rc208-025	138.912	3182.36	108	14514						
rc208-050										
rc208-100										

TABLE 11. Comparison of lower bounds for r-instances.

Problem	NE	NE(2)	NE(4)	E	NE(2)-2PC	Opt
r101-100	1631.15	1631.15	1631.15	1631.15	1634	1637.7
r102-100	1466.6	1466.6	1466.6	1466.6	1466.6	1466.6
r103-100	1203.24	1206.31	1206.54	1206.78	1206.42	1208.7
r104-100	937.31	949.103	955.29	956.973	951.101	971.5
r105-100	1341.19	1346.14	1346.14	1346.14	1349.32	1355.3
r106-100	1212.27	1226.44	1226.44	1226.91	1227.4	1234.6
r107-100	1036.96	1051.84	1052.88	1053.45	1052.94	1064.6
r108-100	891.562	907.162	912.338	913.524	910.617	
r109-100	1097.41	1130.59	1134.28	1134.23	1133.16	1146.9
r110-100	1021.3	1048.48	1055.33	1055.57	1049.94	1068
r111-100	1005.93	1032.03	1034.25	1034.73	1032.07	1048.7
r112-100	892.545	919.192		926.716	922.412	
r201-100	1080.6	1136.22	1140.3	1140.3	1138.65	1143.2
r202-100	935.278					
r203-100		846.489			855.654	
r204-100						
r205-100	838.596	916.572		939.124	922.704	
r206-100		834.57		866.868	840.671	
r207-100						
r208-100						
r209-100	750.431	819.847		841.402	823.733	
r210-100	753.697	849.334		889.370	855.654	
r211-100	650.812	705.806			710.744	

For each instance and each method, we indicate the linear relaxation value (LB), the computation time (CPU), the number of subproblems solved during the column generation process (*No. of iter*) and the number of columns generated (*No. of col*). The column *Gap* gives the gap between the two relaxed values, that is, 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 to 10, for r1-, c1-, rc1-, r2-, c2- and rc2-instances.

The computational results allow us to draw several conclusions. The elementary path version succeed in solving 115 instances out of 168, against 146 for the nonelementary path version. The linear relaxation is tighter in the elementary path version 99 times. The gap between both bounds depends on instances. It is often small (e.g., for c-instances), but might go far beyond 10%, when time windows are large and, consequently, numerous nonelementary routes exist. Furthermore, the elementary path version is able to find directly 68 optimal solutions compared to only 22 for the nonelementary path version.

As expected, the nonelementary path version is generally faster than the elementary path one. Yet, the CPU ratio is often less than 2 (90 times). Furthermore, the elementary path version becomes faster when time windows are larger, because relaxing the elementary condition then results in an extensive increase of the number of feasible routes, that is, feasible states in the dynamic programming scheme. Thus, the elementary path version is faster than the nonelementary one 52 times.

It is interesting to complete these results with complementary tables comparing the impact of the elementary condition on the lower bound value and the impact of other types of improvement. In Tables 11, 12, and 13, we compare several bound values for r-, c-, and rc-instances, respectively. In these tables *NE* stands for Nonelementary and *E* for Elementary. $NE(k)$ indicates that cycles with no more than k arcs are forbidden. $2PC$ means that 2-path cuts are used to improve the quality of the bound. Opt is the optimal solution value. These tables are a compilation of the results presented in this article, complementary results obtained with a faster and not time-restricted version of our algorithm and results from Irnich and Villeneuve [21].

These tables highlight that simply removing cycles of size 2 permits a significant increase of the lower bound value. Even so, the superiority of the elementary-based lower bound on other lower bounds is sometimes notable.

Thus, it is conceivable to expect the efficient solution of the VRPTW using a column generation methodology and our algorithm for the subproblem solution. 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.

By the way, 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

TABLE 12. Comparison of lower bounds for c-instances.

Problem	NE	NE(2)	NE(4)	E	NE(2)-2PC	Opt
c101-100	827.3	827.3	827.3	827.3	827.3	827.3
c102-100	827.3	827.3	827.3	827.3	827.3	827.3
c103-100	826.3	823.3	826.3	826.3	826.3	826.3
c104-100	821.53	822.9	822.9	822.9	822.9	822.9
c105-100	827.3	827.3	827.3	827.3	827.3	827.3
c106-100	827.3	827.3	827.3	827.3	827.3	827.3
c107-100	827.3	827.3	827.3	827.3	827.3	827.3
c108-100	817.352	827.3	827.3	827.3	827.3	827.3
c109-100	809.402	825.64	827.3	827.3	827.3	827.3
c201-100	589.1	589.1	589.1	589.1	589.1	589.1
c202-100	589.1	589.1	589.1	589.1	589.1	589.1
c203-100	585.767	588.7	588.7	588.7	588.7	588.7
c204-100	582.226					
c205-100	582.363	586.4	586.4	586.4	586.4	586.4
c206-100	576.032	585.4	586	586	586	586
c207-100	571.069	581.969		585.8	585.8	585.8
c208-100	570.461	581.767	585.8	585.8	585.8	585.8

Routing Problem with Time Windows (which respectively extend the Selective Traveling Salesman Problem, e.g., Laporte and Martello [24], and the Prize-Collecting Traveling Salesman Problem, e.g., Balas [1]). See Gueguen [17] for a complete description of these applications.

6. CONCLUSION

In this article, 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. First, it reduces the duality gap, compared to the usual approaches that are based on the SPPRC bound. Second, 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 [9], and new resources are introduced to enforce the elementary path constraint, as proposed by Beasley and Christofides [2]. The efficiency of the algorithm is strongly improved 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.

TABLE 13. Comparison of lower bounds for rc-instances.

Problem	NE	NE(2)	NE(4)	E	NE(2)-2PC	Opt
rc101-100	1567.45	1584.09	1584.09	1584.09	1617.4	1619.8
rc102-100	1380.23	1403.65	1405.05	1406.26	1439.55	1457.4
rc103-100	1170.32	1218.5	1223.53	1225.65	1241.72	1258
rc104-100	1052.55	1094.33		1101.81	1112.35	
rc105-100	1453.89	1471.16	1471.93	1471.92	1509.8	1513.7
rc106-100	1248.96	1308.78	1318.8	1318.8	1333.38	
rc107-100	1117.37	1170.69	1182.5	1183.37	1186.43	1207.8
rc108-100	1035.71	1063.01			1097.26	1114.2
rc201-100	1108.69	1240.4	1255.77	1255.94	1253.49	1261.8
rc202-100	888.001	1004.05		1088.08	1013.69	
rc203-100	693.611					
rc204-100						
rc205-100	967.469	1055.52		1147.61	1082.05	
rc206-100	852.817	952.182			982.059	
rc207-100	768.03	866.258		947.313	877.009	
rc208-100						

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 article.

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 article. The authors also thank several anonymous referees for their numerous and very helpful comments and suggestions.

REFERENCES

[1] E. Balas, The prize collecting traveling salesman problem. II: Polyhedral results, Networks 25 (1995), 199–216.

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

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

[4] R.M. Clarke and J.W. Wright, Scheduling of vehicles from a central depot to a number of delivery points, Operat Res 12 (1964), 568–581.

[5] W. Cook and J.L. Rich, A parallel cutting-plane algorithm for the vehicle routing problem with time windows. Technical Report TR99-04, Department of Computational and Applied Mathematics, Rice University, 1999.

[6] J.F. Cordeau, G. Desaulniers, J. Desrosiers, M.M. Solomon, and F. Soumis, "The VRP with time windows," The vehicle routing problem, P. Toth and D. Vigo (Editors), SIAM Monographs on Discrete Mathematics and Applications, 2001, pp. 157–194.

[7] J. Current and M. Marsh, Multiobjective transportation network design and routing problems: Taxonomy and annotation, Eur J Operat Res 65 (1993), 4–19.

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

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

[10] M. Desrochers, J. Desrosiers, and M. Solomon, A new optimization algorithm for the vehicle routing problem with time windows, Operat Res 40 (1992), 342–354.

[11] M. Dror, Note on the complexity of the shortest path models for column generation in VRPTW, Operat Res 42 (1994), 977–978.

[12] I. Dumitrescu and N. Boland, Improved preprocessing, labelling and scaling algorithms for the weight-constrained shortest path problem, Networks 42 (2003), 135–153.

[13] D. Feillet, P. Dejax, and M. Gendreau, Traveling salesman problems with profits, Transport Sci, to appear.

[14] S. Gelinas, M. Desrochers, J. Desrosiers, and M.M. Solomon, A new branching strategy for time constrained routing problems with application to backhauling, Ann Operat Res 61 (1995), 91–109.

[15] M. Gondran and M. Minoux, Graphes et algorithmes, Eyrolles, Paris, 1985.

[16] G.W. Graves, R.D. McBride, I. Gershkoff, D. Anderson, and D. Mahidhara, Flight crew scheduling, Manage Sci 39 (1993), 657–682.

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

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

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

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

[21] S. Irnich and D. Villeneuve, The shortest path problem with k-cycle elimination ($k \geq 3$): Improving a branch and price algorithm for the VRPTW. Technical Report G-2003-55, GERAD, 2003.

[22] B. Kallehauge, J. Larsen, and O.B.G. Madsen, Lagrangean duality applied on vehicle routing with time windows—experimental results. Technical Report IMM-TR-2001-9, IMM, Technical University of Denmark, 2001.

[23] N. Kohl, J. Desrosiers, O.B.G. Madsen, M.M. Solomon, and F. Soumis, 2-Path cuts for the vehicle routing problem with time windows, Transport Sci 33 (1999), 101–116.

[24] G. Laporte and S. Martello, The selective travelling salesman problem, Discrete Appl Math 26 (1990), 193–207.

[25] J. Larsen, Parallelization of the vehicle routing problem with time windows. PhD thesis, Department of Mathematic Modelling, Technical University of Denmark, 1999.

[26] S. Lavoie, M. Minoux, and E. Odier, A new approach for crew pairing problems by column generation with an application to air transportation, Eur J Operat Res 35 (1988), 45–58.

[27] M.E. Lübbecke and J. Desrosiers, Selected topics in column generation, Technical Report G-2002-64, GERAD, 2002.

[28] M.M. Solomon, Vehicle routing and scheduling with time window constraints: Models and algorithms. PhD thesis, Department of Decision Science, University of Pennsylvania, 1983.

[29] A. Warburton, Approximation of pareto optima in multiple-objective, shortest-path problems, Operat Res 35 (1987), 70–79.

[30] M. Ziegelmann, Constrained shortest paths and related problems. PhD thesis, Universität des Saarlandes, 2001.