New Refinements for the Solution of Vehicle Routing Problems with Column Generation

Dominique Feillet* Michel Gendreau† Louis-Martin Rousseau†

*Laboratoire d'Informatique d'Avignon
339 Chemin des Meinajariés
BP 1228, 84911 Avignon France
dominique.feillet@lia.univ-avignon.fr

†Center for Research on Transportation, Montreal University
C.P. 6128, Succursale Centre-Ville
Montreal, H3C 3J7 Canada
michelg@crt.umontreal.ca

Introduction

Vehicle Routing Problems (VRP) are widely present in today's industries, ranging from distribution problems to fleet management. They account for a significant portion of the operational costs of many companies. If most real instances of VRPTW are solved with heuristic methods, the desire to produce optimal or near optimal solutions has always motivated the research in the area of exact methods.

The Vehicle Routing Problem (VRP) can be described as follows: given a set of customers, a set of vehicles, and a depot, find a set of routes of minimal length, starting and ending at the depot, such that each customer is visited by exactly one vehicle. Each customer having a specific demand, there are usually capacity constraints on the load that can be carried by a vehicle. In addition, a maximum amount of time that can be spent on the road is sometimes considered. The time window variant of the problem (VRPTW) imposes the additional constraint that each customer must be visited during a specified time interval. One can wait in case of early arrival, but late arrival is not permitted.

Section 1 will briefly review the column generation approach for the VRPTW. Section 2 will present an overview of the different ideas we propose. Experimental results evaluating the techniques are provided in section 3.

1 Column Generation

Column generation was introduced by Dantzig and Wolfe [2] for the solution of linear programs with decomposable structures. It has been applied to many problems with success and has become a leading optimization technique to solve Routing and Scheduling Problems [4, 1].

In the first application of column generation to the field of Vehicle Routing Problems with Time Windows, presented by Desrochers *et al.* [3], the basic idea was to decompose the problem into sets of customers visited by the same vehicle (routes) and to select the optimal set of routes between all possible ones. Letting r be a feasible route in the original graph (which contains N customers), R be the set of all possible routes r, c_r be the cost of visiting all the customers in r, $A = (a_{ir})$ be a Boolean matrix expressing the presence of a particular customer (denoted by index $i \in \{1..N\}$) in route r, and x_r be a Boolean variable specifying whether the route r is chosen ($x_r = 1$) or not ($x_r = 0$), the Set Partitioning formulation is defined as (S):

$$\min \sum_{r \in R} c_r x_r \tag{1}$$

$$s.t \sum_{r \in R} a_{ir} x_r \quad = 1 \quad \forall i \in \{1..N\} \tag{2}$$

$$x \in \{0,1\}^{|R|} \tag{3}$$

This formulation however poses some problems. Firstly, since it is impractical to construct and to store the set R because of its very large size, it is usual to work with a partial set R' that is enriched iteratively by solving a subproblem. Secondly, the Set Partitioning formulation is difficult to solve when R' is small and it allows negative dual values which can be problematic for the subproblem (a negative dual means there is a negative marginal cost to visit a node...). This is why, in general, the following Set Covering formulation is used instead as a Master Problem (M):

$$\min \sum_{r \in R'} c_r x_r \tag{4}$$

$$s.t \sum_{r \in R'} a_{ir} x_r \quad \geq 1 \quad \forall i \in \{1..N\} \tag{5}$$

$$x \geq 0 \tag{6}$$

1.1 Subproblem

To enrich R', it is necessary to find new routes (x variables) that offer a better way to visit the customers they contain, that is, routes with a negative reduced cost. The reduced cost of a route is calculated by replacing the cost of an arc (the distance between two customers) d_{ij} by the reduced cost of that arc $c_{ij} = d_{ij} - \lambda_i$, where λ_i is the dual value associated with the covering constraint (5) of customer i. The dual value associated with a customer can be interpreted as the marginal cost of visiting that customer in the current optimal solution (given R'). The objective of the subproblem is then the identification of a negative reduced cost path, i.e., a path for which the sum of the traveled distance is inferior to the sum of the marginal costs (dual values). Such a path represents a new and better way to visit the customers it serves (and a x variable with a negative reduced cost).

Here we give a brief description of the algorithm used to solve the subproblem. More information on this algorithm can be found in Desrochers *et al.* [3] for example. The algorithm is an extension of the classical Bellman's algorithm. The principle is to associate with each possible partial path a label and to extend these labels checking resource constraints until the best feasible paths are obtained. Dominance rules are used to compare partial paths arriving at a same location. But, unlike Bellman's algorithm when no resources are considered, each vertex of the graph can maintain a large number of labels since the comparison of two labels takes account of their consumption level for each resource.

1.2 Search for Integer Solutions

The optimal solution of (M) has been identified when there exists no more negative reduced cost path. This solution can however be fractional, since (M) is a relaxation of (S), and thus does not represent the optimal solution of (S) but rather a lower bound on it. If this is the case, it is necessary to start a branching scheme in order to identify an integer solution. More information on this part of the column generation framework shall be provided in the complete version of the paper.

2 New Acceleration Techniques

The contribution of this paper lies in the following improvements that were incorporated in the subproblem solution algorithm. Note that contrary to usual implementations of column generation for routing problems, we only consider elementary routes in R, that is routes where

customers are never visited more than once. Details on this implementation can be found in Feillet *et al.* [5]. But, although these techniques were implemented in the context of elementary shortest paths, they all could be adapted to be used in a non-elementary context.

2.1 Limited Discrepancy Search

Limited Discrepancy Search (LDS), introduced by Harvey and Ginsberg [6], is a very well known tree search method in Constraint Programming (CP). Given a heuristic that ranks all the possible branches at a given node of a search arborescence, a *discrepancy* is a branching decision that does not follow the heuristic recommendation. LDS *trusts* the heuristic by first exploring the branches which corresponds to lowest discrepancy and by setting a limit on the maximum number of discrepancy allowed. Limited Discrepancy Search is thus a very effective technique which allows identifying rapidly good solutions by exploring the most promising region of the solution space.

We embed the concept of LDS in Dynamic Programming (DP) to efficiently generate the most promising paths of negative value. LDS is implemented by adding a label to every node which indicates the number of discrepancies performed in order to generate that label. For each node of the original VRPTW, we define a set of good neighbors and a set of bad neighbors according to distances and time considerations. A discrepancy is then counted each time a label is extended along an arc connecting a node to one of its bad neighbors. If the extension of a label to a particular node causes the discrepancy level of that label to exceed the current global maximum discrepancy level, the label is simply not extended.

By using LDS and by limiting the number of path that are added to R', the column generation iterations are rapidly executed until almost no valid negative paths remain (a few iterations before the process terminates). The maximum discrepancy level is progressively increased when the subproblem fails to find new columns in order to ensure optimality.

2.2 Label Loading and Meta Extensions

The motivation behind these techniques is to use, when possible, the information about the "good" paths that have been previously identified. Label Loading (LL) consists of adding a set of labels to the graph before the DP search process is undertaken. This is very simple and has presumably been implemented in other DP algorithm addressing similar problems. Our implementation consists of selecting in R' all routes r for which $x_r > 0$, since these routes are used in the optimal covering solution. We then traverse each of these routes while generating the label associated with the visit of each node. All the labels thus generated are added to the DP graph before we start looking for new routes.

The Meta Extension (ME) operator is used to obtain the complementary effect of Loading Labels. While traversing each of the routes previously selected, we also add new *metanodes* to the original graph. These metanodes correspond to the remaining path from their associated original node to the destination depot. For instance, when traversing route $Depot - A - B - C - D - E - Depot$, the metanode we associate with C is the equivalent (in cost and resources) to the remaining subpath $C - D - E - Depot$. When a label l is extended to a metanode m,

the path created by connecting of the segments from the l's history and m's subpath instantly yields a complete path. Furthermore, the cost value and resource consumption of this new path can be ontained in constant time, since all the necessary information was computed once for each metanode at the moment of its creation.

These two techniques, when used together, allow to rapidly identify small variants of the best routes in R'. Operators like node insertion, node deletion, and path crossing (connecting the end of one route to the beginning of another) can be obtained with only a few label extensions.

2.3 Label Elimination

When the resource limits are not constraining, a very high number of labels can be generated during the search for negative reduced costs paths. If we could determine which label can never be extended to the final depot with a negative cost, we could probably eliminate a significant proportion of these labels.

What we propose is to compute a very simple lower bound on the costs of all paths which can be generated with a given label. This bound can be computed after each label extension in order to establish whether the label can still yield a negative cost path. The bound has to be chosen to balance the quality of the information it provides with the time it takes to be computed. We are now experimenting with two knapsack bounds based on the capacity and time resources. The linear relaxation of these knapsacks can be computed in linear time, which is acceptable in the context of an Elementary Shortest Path algorithm since linear computations are already performed after each label extension.

Preliminary results show that this technique can be very efficient in eliminating a large amount of labels (sometimes over 50%) but the execution times tend to diminish only by a small factor. However, we are still actively pursuing this research direction.

3 Experimental Results

We evaluated the performance of the techniques we have proposed on several different VRPTW instances. But before looking at the figures from the experimental results we shall briefly describe the instances used and our implementation of the column generation framework.

3.1 Benchmarks

We have tested the proposed method on the well-known Solomon instances [8]. The geographical data are randomly generated in problem sets R1, clustered in problem sets C1, and a mix of random and clustered structures in problem sets RC1. The customer coordinates are identical for all instances within one type (i.e., R, C and RC). The instances differ with respect to the width of the time windows. Some have very tight time windows, while others have time windows which are hardly constraining. Each instance contains 100 customers.

In the data sets, the distance matrix is not explicitly stated, but customer locations are given.

Problem Class	Nb Solved	Without Improvements		With Improvements	
		Avg Time	Avg Iter.	Avg Time	Avg Iter
R1	9	137.2	36.9	46.2	86.7
C1	8	105.5	26.4	18.6	39.3
RC1	7	336.3	37.1	212.8	98.6
R2	3	590.7	66.3	162.3	149.0
C2	3	1615.6	135.6	150.1	102.0
RC2	4	507.1	66.0	144.3	151.0

Table 1: Iteration count and CPU time comparison over problems solved by both method.

Euclidean distances between these customers are calculated with one decimal point, to allow comparison with other published methods.

3.2 Implementation of Column Generation

The column generation framework used to evaluate the proposed methods is quite straightforward since the Restricted Master Problem is the one described by (4)-(6). Columns are generated using resource constrained elementary shortest path algorithm based on dynamic programming (Feillet *et al.* [5]). At each iteration of the subproblem, the number of labels extended with a negative cost to the destination node is limited to 500. The maximum allowed time to find a solution was set to 3600 seconds.

The results reported here concern only the linear relaxation of the path based model of the VRPTW as we are now working on completing the branch and price framework. Since the accelerations we have proposed here concern the column generation process, the search for integer solution should benefit from the same improvement as those observed during the lower bound computation. As we should have completed the whole framework in a few months, we intend to present results on the complete solution process at Tristan IV.

Computational experiments were carried out on a 2.8 GHz Pentium 4 with 1.5 Gb of RAM. The Master Problem was modeled and solved with Cplex 7.5 leaving all parameters to their default values. The overall process is stabilized with the techniques presented in [7].

3.3 Results and Discussion

In table 1 we compare the CPU time and the number of iterations needed to obtain the LP bound of the Master Problem. Results are presented for two versions of the shortest path algorithm, where the first one is the basic version described in [5] and the second one which uses the proposed improvements. The average values are computed only on the problems that could be solved by both versions in less than 3600 seconds. When using the improvements, more iterations are needed to terminate the column generation process, but the CPU time is significantly decreased. The benefits of these techniques thus seem to be considerable.

Furthermore, the proposed methods allow solving more problems within the one hour time

Problem	LP bound	CPU Time	Iterations
r104	956.973	858.7	168
r108	913.524	1553.7	133
r112	926.716	1403.4	144
rc104	1101.810	1983.3	132
r206	866.868	2202.5	146
r209	841.402	591.8	176
r210	889.370	3425.0	206
c202	589.100	2078.8	88
c207	585.8	184.7	91
c208	585.8	100.6	75
rc207	947.313	2080.3	179

Table 2: Bounds obtained with refined algorithm only

limit. Table 2 reports the problems that were solved only with the improved shortest path algorithm. Out of the 56 test instances, we could not identify the elementary lower bound of 1 problem in the first class of instances and 10 problems in the seconde one.

4 Conclusion

We have presented three techniques which accelerate dynamic programming algorithm addressing the Shortest Path Problem with Resource Constraint in a column generation context. The Limited Discrepancy Search allows to rapidly execute the first iterations of column generation and to concentrate the effort on the last iterations. Label Loading and Meta Extension are simple techniques which allow transforming the traditional label extension procedure into more powerful local search operators. Finally by computing a lower bound after each extension we are able to identify and remove a large number of labels which can be proved worthless.

This project is subject to ongoing research as we are now building and testing the global branch and price algorithm that will enable the identification of integer (and thus feasible) solution to the VRPTW. Since column generation is performed throughout the branch and price, the methods presented here will be extremely useful in building an efficient framework.

References

[1] C. Barnhart, E.L. Johnson, G.L. Nemhauser, M.W.P. Savelsbergh, and P.H. Vance. Branch-and-Price: Column Generation for Huge Integer Programs. *Operations Research*, 46:316–329, 1998.

[2] G.B. Dantzig and P. Wolfe. Decomposition principles for linear programs. *Operations Research*, 8:101–111, 1960.

[3] M. Desrochers, J. Desrosiers, and M.M. Solomon. A New Optimisation Algorithm for the Vehicle Routing Problem with Time Windows. *Operations Research*, 40:342–354, 1992.

[4] J. Desrosiers, Y. Dumas, M.M. Solomon, and F. Soumis. Time Constrained Routing and Scheduling. In M.O. Ball, T.L. Magnanti, C.L. Monma, and Nemhauser G.L., editors, *Network Routing*, volume 8 of *Handbooks in Operations Research and Management Science*, pages 35–139. North-Holland, Amsterdam, 1995.

[5] D. Feillet, P. Dejax, M. Gendreau, and C. Gueguen. An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. Technical report, Laboratoire Informatique d'Avignon, 2003.

[6] W. Harvey and M. Ginsberg. Limited Discrepancy Search. In *Proc. of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95)*, pages 607–615, Montréal, Canada, 1995. Morgan Kaufmann.

[7] L.-M. Rousseau, M. Gendreau, and D. Feillet. Interior Point Stabilization for Column Generation. Technical report, Centre de recherche sur les transports, 2003.

[8] M. M. Solomon. Algorithms for the Vehicle Routing and Scheduling Problem with Time Window Constraints. *Operations Research*, 35:254–265, 1987.