Diversification and reoptimization procedures in column generation for the resolution of the acyclic vehicle routing problem with time windows

Touati Nora* Lucas Létocart* Anass Nagih*

*LIPN, UMR 7030 CNRS, Université Paris 13,
99 avenue Jean-Baptiste Clément 93430, Villetaneuse, France

*LITA, Université Paul Verlaine, Ile du Saulcy 57045 Metz Cedex 1, France

Abstract

This paper focuses on accelerating strategies in a Column Generation (CG) algorithm. In order to decrease the total number of generated columns and then, master problems resolution time, pricing problems solutions are made of task-disjoint columns. This can be achieved by diversification methods. Another way to improve CG computing time is to use reoptimization approaches to solve efficiently the pricing problems. We show in this work that diversification approaches are more efficient when applied on the first iterations to build efficiently a good approximation of pricing problems convex hull and that reoptimization methods are more efficient when applied on the last iterations when the dual variables are close. We combine a diversification technique and a reoptimization procedure in a CG scheme to improve the global resolution time. This study is validated on the Vehicle Routing Problem with Time Windows (VRPTW), defined on acyclic networks.

Keywords: *Diversification; Reoptimization; Column generation; Vehicle routing problem with time windows.*

1 Introduction

A well known strategy to reduce the number of iterations in CG, is to return to the Master Problem (MP) many negative marginal cost columns at each iteration (minimization problem case). Generally, this intensification typically overloads the MP, while the final optimal base contains a very restricted subset of all the generated columns. Diversification methods which consist of computing complementary columns at each iteration of CG, permit to decrease the number of generated columns by characterizing quickly a good approximation of the Pricing Problem's (PP) convex hull. This approach is then more efficient on the first iterations [6, 8].

Another way to improve the CG algorithm is to solve efficiently the PPs, known to be very greedy on computing time in many applications. We interest for this purpose to reoptimization techniques, which consist of using some informations about the PP's resolution at an iteration k, for solving efficiently the PP at iteration $k+1$ [7, 8].

This work focuses on the combination of diversification and reoptimization techniques in CG. We apply here a diversification method on the first iterations, and a reoptimization procedure on the last ones. The relevance of this approach is demonstrated by computational experiments on Solomon and on randomly generated instances of the Acyclic VRPTW (AVRPTW).

2 Vehicle routing problem with time windows

The VRPTW ([2], [3], [5]) involves the design of a set of minimum cost routes, originating and terminating at a central depot, and visiting each costumer once, for a fleet of identical vehicles. The oriented network used by vehicles is defined by $\mathcal{G} = (\mathcal{V}, \mathcal{A})$, where $\mathcal{V}$ is the set of nodes composed of the set of costumers $\mathcal{N}$, a source 's' and a destination 't' representing exit and entrance to the depot respectively, and $\mathcal{A}$ the set of arcs connecting the customers and the depot. With each arc $(i, j) \in \mathcal{A}$ is associated a positive duration t_{ij} and a cost c_{ij}. Each vehicle has a given capacity Q and each customer i has a demand d_i, $i \in \mathcal{N}$. For each customer i, the start of the service must be within a given time window $[a_i, b_i]$. A vehicle can wait in case of early arrival. The VRPTW can be formulated as a set partitioning problem, its linear relaxation is called the master problem and given as follows :

$$(MP_{VRPTW}) \qquad \min \textstyle\sum_{r\in\mathcal{R}} c_r\lambda_r, \qquad \sum_{r\in\mathcal{R}} \delta_{ir}\lambda_r = 1, \forall i \in \mathcal{N}, \;\; \lambda_r \geq 0, \forall r \in \mathcal{R},$$

where $\mathcal{R}$ is the set of all feasible paths, i.e. paths going from 's' to 't' and satisfying capacity and time windows restrictions, c_r is the cost of path r and δ_{ir} is 1 if path r visits node i and 0 otherwise. The PP is equivalent to a Shortest Path Problem with Time Windows and Capacity Constraints (SPPTWCC) and formulated as follows:

$$(SPPTW) \begin{cases} \min & \sum_{(i,j)\in\mathcal{A}}(c_{ij} - \pi_i)x_{ij} & \\ & \sum_{j\in\mathcal{N}} x_{sj} = \sum_{j\in\mathcal{N}} x_{jt} = 1 & \\ & \sum_{j\in\mathcal{N}} x_{ij} - \sum_{j\in\mathcal{N}} x_{ji} = 0, & \forall i \in \mathcal{N} \\ & \sum_{j\in\mathcal{N}} d_j \sum_{i\in\mathcal{N}\cup\{s\}} x_{ij} \leq Q & \\ & x_{ij} \in \{0, 1\}, & \forall (i, j) \in \mathcal{A} \\ & x_{ij}(T_i + t_{ij} - T_j) \leq 0, & \forall (i, j) \in \mathcal{A} \\ & T_j \in [a_j, b_j], & \forall j \in \mathcal{N} \\ & T_j \in \mathbb{N}, & \forall j \in \mathcal{N} \end{cases}$$

where $x_{ij}, (i, j) \in \mathcal{A}$ are binary flow variables taking the value 1 when arc (i, j) is used by a vehicle, $T_i, i \in \mathcal{N}$ are time variables associated with the starting of the service at node i, and π_i is the dual variable associated with the i-th constraint of the MP. The standard approach to solve the SPPTW in practice is based on the dynamic programming method and has a pseudo-polynomial complexity. The principle is to associate with each possible partial path a label indicating the cumulated cost, time and demand, and to eliminate labels with the help of dominance rules. We consider here a label correcting algorithm, where nodes are repeatedly treated and their labels extended [4]. Solving the SPPTW with a dynamic programming approach is closely related to solving multi-objective shortest path problems, the aim is to generate efficient paths (i.e., Pareto optimal paths).

The main goal of our study is to evaluate the impact of two accelerating techniques of the column generation algorithm. Our study focuses on a particular version of the vehicle routing problem with time windows, based on acyclic networks, nevertheless many routing problems in practice are modeled by acyclic networks, particularly when the width of time windows is lower than the intertask time. It is the case in rail transportation problems for example.

Vehicle routing problem with time windows test instances

We consider in our experimentations two different types of AVRPTW instances. Some Solomon test problems from which we can extract acyclic networks ($C101\_25$ (25 nodes), $C101\_50$ (50 nodes), $C101$ (100 nodes), $C1\_2\_1$ (200 nodes)), and 10 AVRPTW instances uniformly randomly generated for each size: G_{100} (100 nodes), G_{120} (120 nodes), G_{140} (140 nodes) and G_{160} (160 nodes).

3 Diversification in a column generation algorithm

It is well known that the intensification allows to decrease the number of iteration of CG. Experimentations on AVRPTW instances described in section 2, reveal two interesting phenomena. The average percentage

of suboptimal columns in the final optimal base is higher than 93% and 99% of the generated columns do not belong to the final optimal base.

To avoid the fast and useless increase of MPs size, we classically restrict to the generation of the k best solutions, k is a parameter expressed in % in what follows. We call this procedure k%_Intensified Column Generation (k%_ICG). Intensified Column Generation (ICG) correspond to $k = 100$. Experimentations on the AVRPTW [6] showed that in average, the number of columns increases when k increases, whereas the number of iterations and the global resolution time decrease. Even if the add of columns with good reduced cost decreases the global number of generated columns, it increases considerably the number of iterations and is not enough to improve the global computing time. Instead of select columns based on the reduced cost criterion, we interest in what follows to columns structure based criterion.

3.1 Suboptimal columns characteristics

The previous experimental results show that suboptimal columns of best reduced cost are neighbor compared to the optimal column obtained, i.e. they covered almost the same tasks. We expose bellow the 20-th columns of best reduced cost, generated for a Solomon test instance of 25 costumers. The first column represent the path's cost and the others, the sequence of path's nodes.

```
ITERATION 1
-1140.52 i0 i3 i7 i8 i10 i11 i9 i6 i4 i2 i1 i26
-1133.41 i0 i3 i7 i8 i10 i11 i9 i6 i23 i22 i21 i26
-1103.58 i0 i3 i7 i8 i10 i11 i9 i6 i23 i22 i1 i26
-1086.77 i0 i3 i7 i19 i10 i11 i9 i6 i23 i22 i21 i26
-1056.95 i0 i3 i7 i19 i10 i11 i9 i6 i23 i22 i1 i26
-1043.13 i0 i3 i7 i8 i10 i11 i9 i6 i4 i1 i26
-1040.59 i0 i3 i7 i8 i10 i11 i9 i6 i4 i2 i26
-1036.05 i0 i3 i7 i8 i10 i11 i9 i23 i22 i21 i26
-1033.44 i0 i3 i7 i8 i10 i11 i9 i6 i23 i22 i26
-1031.38 i0 i3 i7 i8 i10 i11 i9 i6 i4 i22 i26
-999.81 i0 i3 i7 i8 i10 i11 i9 i6 i12 i22 i26
-996.493 i0 i3 i7 i19 i10 i11 i9 i6 i4 i1 i26
-993.954 i0 i3 i7 i19 i10 i11 i9 i6 i4 i2 i26
-989.421 i0 i3 i7 i19 i10 i11 i9 i23 i22 i21 i26
-986.806 i0 i3 i7 i19 i10 i11 i9 i6 i23 i22 i26
-984.749 i0 i3 i7 i19 i10 i11 i9 i6 i4 i22 i26
-946.696 i0 i3 i7 i8 i10 i11 i9 i6 i4 i26
-944.56 i0 i3 i7 i10 i11 i9 i6 i4 i1 i26
-942.02 i0 i3 i7 i10 i11 i9 i6 i4 i2 i26
-937.487 i0 i3 i7 i10 i11 i9 i23 i22 i21 i26
...
```

In the PP's feasible solutions space, these columns are close compared to the PP's optimal solution, so, they can not improve the MP's feasible convex hull approximation. In the dual space, cuts corresponding to close columns are redundant for the current dual function approximation.

Thus, the addition of very close columns can be useless. The ICG needs to generate less columns, but good ones, for improving significantly the MP's convex hull approximation (and the dual function approximation). For this purpose, instead of selecting columns according to the reduced cost criterion, we interest in column's structure criterion. We look for complementary columns (i.e. columns don't having joint nodes) characteristics.

It is known that the add of complementary columns (resp. orthogonal cuts) to the MP permits to widen its convex hull (resp. the dual function) approximation [1, 9]. It is showed in [6] that this approach is more efficient on the first iterations when it is necessary to compute quickly a good primal (or dual) solution. On the last iterations of CG, the MP's convex hull is well approached and it is preferable in this case to generate columns to complete the current basis in order to obtain an optimal solution.

3.2 Diversification by resolution procedure and experimentations

Diversification procedures consist of inserting at each iteration of CG, a set of complementary columns to the MP. We consider here the CG with Diversification by Resolution (CGDR) [6], which consists at each iteration of CG to solve a sequence of SPPTWCC, each one providing a path. After solving the l-th problem for a set of nodes $\mathcal{N}^l$, the nodes S^l belonging to the optimal solution obtained are removed from $\mathcal{N}^l$ to form a set of nodes $\mathcal{N}^{l+1} = \mathcal{N}^l \setminus S^l$ which will be used for the $(l+1)$-th problem.

We applied this procedure on AVRPTW instances described in section 2. Diversification by resolution is applied on the first iterations, when $\frac{v(MP^k)-v(MP^{k-1})}{v(MP^k)} \geq \varepsilon$, where $v(MP^k)$ is the MP's value at iteration

k. The intensification is used on the other iterations. All results reported in table 3 for each randomly generated class size are average values over 10 test instances.

Solomon's	$C101\_25$				$C101\_50$				$C101$				$C1\_2\_1$			
instances	nbIt	nbC	GRT	nD	nbI	nbC	GRT	nD	nbIt	nbC	GRT	nD	nbIt	nbC	GRT	nD
20%_ICG	51	153	4,3"	0	127	647	73,1"	0	170	2 008	15,2'	0	286	10 768	438,6'	0
50%_ICG	30	201	2,8"	0	61	823	32,7"	0	105	2 154	9,6'	0	250	11 020	197,1'	0
ICG	24	246	2,1"	0	55	1 003	31,7"	0	63	2 724	5,0'	0	163	11 460	150,9'	0
CGDR	35	251	3,0"	3	83	956	51,8"	4	75	2 646	2,1'	2	114	10 141	102,2'	5
Generated	G_{100}				G_{120}				G_{140}				G_{160}			
instances	nbIt	nbC	GRT	nD	nbIt	nbC	GRT	nD	nbIt	nbC	GRT	nD	nbIt	nbC	GRT	nD
20%_ICG	69	6 263	10,3'	0	99	7 864	32,3'	0	97	10 169	43,5'	0	72	10 409	18,0'	0
50%_ICG	36	11 406	9,8'	0	48	13 663	31,2'	0	49	18 800	40,1'	0	42	17 980	20,0'	0
ICG	27	16 960	12,5'	0	26	19 013	39,1'	0	25	25 259	35,5'	0	27	23 864	20,4'	0
CGDR	40	2 589	5,2'	22	50	2 579	17,5'	28	55	5 370	15,4'	37	48	1 298	6,8'	31

nbIt: global number of iterations.
nbC: number of generated columns.
GRT: global resolution time (') minutes and (") seconds).
nD: number of iterations with diversification.
$\varepsilon = 0.001$.

Table 1: Generation of complementary solutions

Table 1 shows that diversification permits to decrease the number of computed columns without increasing considerably the number of iterations (compared to 20%_ICG and 50%_ICG). We generate with CGDR in average 61%, 69% and 79% of columns generated by ICG, 50%_ICG and 20%_ICG procedures respectively, this decreases MPs size and their resolution time. Diversification reduces the ICG resolution time by 39%.

4 A reoptimization technique in CG

Consider the following problems:

$$(P) : \{\max cx, \quad x \in X \subset \mathbb{N}^n\} \quad \text{and} \quad (P') : \{\max c'x, \quad x \in X' \subset \mathbb{N}^n\},$$

where $c, c' \in \mathbb{R}^n, n \in \mathbb{N}$. Reoptimization consists of studying possibilities of an efficient resolution of (P') using some informations about the resolution of (P).

In the CG process, only some PP's objective coefficients change, due to the update of MP's dual solutions. So, reoptimization in this case can be used. The principle is to retain some informations from the PP's resolution at an iteration k, to be used at iteration $k+1$, for accelerating the PP's resolution.

We denote by $\mathcal{E}_j^k$ the set of efficient labels of node $j \in \mathcal{V}$ at iteration k of CG. The set $\mathcal{E}_j^k$ permits to build a Pareto frontier $\mathcal{P}_j^k$ representing a lower bound on the cost of all feasible labels at node j.

Master problem's resolution at iteration $k+1$, gives new dual variables values. Labels in $\mathcal{E}_j^k$ are still feasible at this iteration but their cost change, they define a primal function representing an upper bound on costs of labels in $\mathcal{E}_j^{k+1}$ as function of time. Then, the Pareto optimal frontier computed at iteration k allows to define a primal function at iteration $k+1$. To compute the set $\mathcal{E}_j^{k+1}$, it is enough to calculate all labels dominating those in $\mathcal{E}_j^k$.

Labels computed at iteration $k+1$ belong to the space between the primal function and $\mathcal{P}_j^{k+1}$, called treatment zones. We reduce then the space search of efficient labels, the number of labels computed and the dominance procedure time. The smaller are the treatment zones, the smaller is the space search of new efficient labels and more reoptimization is efficient.

It is frequently observed in a CG process, that MP's values evolve slightly on the last iterations, this phenomenon is called the long tail effect, so, the dual variables are also close, and the change of arcs costs is weak. The treatment zones on these iterations can be small and reoptimization is more efficient.

We tested this approach on AVRPTW instances. ICG is used until the inequality $\frac{v(MP)^k - v(MP)^{k-1}}{v(MP)^k} \leq \varepsilon$ is satisfied, next, the reoptimization technique is used. We call this procedure CG with Reoptimization (GCReopt).

Solomon's	$C101\_25$			$C101\_50$			$C101$			$C1\_2\_1$		
instances	GRT	nbLb	ItR/ItG	GRT	nbLb	ItR/ItG	GRT	nbLb	ItR/ItG	GRT	nbLb	ItR/ItG
ICG	2,1"	9 112	0/24	31,7"	86 073	0/55	5,0'	384 810	0/63	150,9'	4 262 215	0/163
GCReoptP	1,2"	3 968	18/24	29,0"	18 178	48/55	2,1'	118 292	52/63	76,8'	1 148 708	114/163
Generated	G_{100}			G_{120}			G_{140}			G_{160}		
instances	GRT	nbLb	ItR/ItG	GRT	nbLb	ItR/ItG	GRT	nbLb	ItR/ItG	GRT	nbLb	ItR/ItG
ICG	12,5'	446 080	0/27	39,1'	5 841 413	0/26	35,5'	454 729	0/25	20,4'	475 701	0/27
GCReoptP	8,9'	315 241	8/27	21,2'	611 876	9/26	21,1'	405 266	7/25	17,3'	423 800	4/27

GRT: global resolution time (') minutes and (") seconds).
nbLb: number of treated labels.
ItG: global number of iterations.
ItR: number of iterations with reoptimization.
$\varepsilon = 0.001$.

Table 2: Reoptimization in a CG algorithm

Experimental results reported on table 2 show that in average, GCReopt decrease the number of computed labels by 53% compared to ICG. This is due to the use at each iteration of efficient labels computed at the previous iteration, allowing the construction of primal functions which restrict the space search of new efficient labels. Pricing problems resolution time is then significantly decreased. The average gain of computing time obtained on the global resolution time of the ICG algorithm is 36%. We observe that the gain obtained by the GCReopt on the ICG global computing time is not as important as the gain on the number of treated labels, this is due to the overcost of primal functions construction.

5 Diversification and reoptimization in a CG algorithm

In the CG algorithm we can distinguish two important phases. The first one, is the begin of the process, where the MP is poor in informations. Many methods in literature propose techniques to enrich the MP (start the CG with a pool of columns generated heuristically or with a good dual solution ...). We interest here on using complementary solutions to provide good informations for the MP by characterizing efficiently a good approximation of its convex hull. The second one is the end of the process, where we frequently observe a weak evolution of MPs values. On the last iterations the dual variables values are neighbor, so, reoptimization methods can be efficient to reduce the resolution time of the PP.

We showed that the ICG is improved by using diversification or reoptimization techniques. We aim to study the impact of their combination on CG performances. We use diversification on the first iterations until the inequality $\frac{v(MP)^k - v(MP)^{k-1}}{v(MP)^k} >= \varepsilon$ is satisfied and reoptimization on the other iterations. We call this procedure Column Generation with Diversification and Reoptimization (CGDReopt). Table 3 presents a comparative study between ICG, CGDR, CGReopt and CGDReopt procedures, by experimentations on the test instances described in section 2.

instance	$C101\_25$	$C101\_50$	$C101$	$C1\_2\_1$	G_100	G_120	G_140	G_160
ICG	2,1"	31,7"	5,0'	150,9'	12,5'	39,1'	35,5'	20,4'
CGDR	1,7"	20,8'	2,1'	102,2'	5,2'	17,5'	15,4'	6,8'
CGReopt	1,2"	29,0"	2,1'	76,8'	8,9'	21,2'	21,1'	17,3'
CGDReopt	1,3"	23,5"	2,0'	71,6'	4,3'	10,6'	15,6'	10,4'

$\varepsilon = 0.001$

Table 3: Combination of diversification and reoptimization in CG

The ICG procedure computing time is improved on one hand by using diversification on the first iterations where the gain obtained is in average 48% and on another hand by using reoptimization on the last iterations, this decrease the ICG resolution time by 34% in average. The gain on the global computing

time is related to the number of iterations made with diversification and reoptimization respectively.

As expected, the use of both these two methods decreases significantly the resolution time of the ICG procedure, the average gain obtained on computing time is respectively.

For our experimentations, the compromise between the cost of treatment of the accelerating procedure (the computation of complementary columns for CGDR and the construction of primal functions in CGReopt) and the gain on the global resolution time (gain on the number of generated column in CGDR and the gain on the number of treated labels for CGReopt) is better for the CGDR procedure than CGReopt.

6 Conclusion

Diversification methods give a very interesting gain on the global number of generated columns in GC by building a good approximation of PP feasible domain. It reduces the MP's size and the global resolution time. These approaches are more interesting when used on the first iterations where the PP's convex hull approximation is poor.

Reoptimization approaches are used successfully to improve PP's resolution time in CG. We present a reoptimization algorithm in CG for the resolution of AVRPTW. The principle is to use efficient labels computed at an iteration k to construct a primal function for reducing the space search of new efficient labels at iteration $k+1$. This reduces the number of treated labels and dominance time.

We combine in this work these two alternatives in CG by applying a diversification method on the first iterations and a reoptimization procedure on the last ones. Experimental results on AVRPTW instances show that the gain on computing time obtained by this combination is much more important than that obtained by the two approaches used independently.

References

[1] M. Desrochers and F. Soumis, *A reoptimization algorithm for the shortest path problem with time windows*, European Journal of Operational Research, **35**, (1988) pp. 242–254.

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

[3] G. Desrosiers, F. Soumis and M. Desrochers, *Routing with time windows by CG* Networks, **14**, (1984) pp. 545–565.

[4] M. Desrochers, *An algorithm for the schortest path problem with ressource constraintes*, Les cahiers du GERAD, (1988) G-88-27

[5] M.L. Fisher, K.O. Jornsten and O.B.G. Madsen, *Vehicle routing with time windows : Two optimization algorithms*, Operations Research, **45(3)**, (1997) pp. 488–492.

[6] N. Touati, L. Létocart and A. Nagih, *Solutions diversification in a CG scheme*, Technical report, LIPN, (2008).

[7] N. Touati, L. Létocart and A. Nagih, *Reoptimization techniques in a CG scheme*, Technical report, LIPN, (2008).

[8] N. Touati, *Performances improvement of the column generation algorithm: Application on the vehicle routing problem with time windows*, Thesis, Computer Science lab of the Paris 13 University (2008).

[9] F. Vanderbeck, *Decomposition and Column Generation for Integer Programs*, PhD thesis, UCL (1994).