Solutions diversification in a column generation algorithm

N. Touati [a] L. Létocart [a] A. Nagih [b]

[a] *LIPN, UMR 7030 CNRS, Institut Galilée - Université Paris 13, 99 avenue Jean-Baptiste Clément 93430 Villetaneuse, France.*

[b] *LITA, Université Paul Verlaine, Ile du Saulcy 57045 Metz Cedex 1, France.*

Abstract

Column generation algorithms have been specially designed for solving mathematical programs with a huge number of variables. Unfortunately, this method suffers from slow convergence that limits its efficiency and usability. Several accelerating approaches are proposed in the literature such as stabilization-based techniques. A more classical approach, known as "intensification", consists in inserting a set of columns instead of only the best one. Unfortunately, this intensification typically overloads the master problem, and generates a huge number of useless variables. This article covers some characteristics of the generated columns from theoretical and experimental points of view. Two selection criteria are compared. The first one is based on column reduced cost and the second on column structure. We conclude our study with computational experiments on two kinds of problems: the acyclic vehicle routing problem with time windows and the one-dimensional cutting stock problem.

Key words: column generation; diversification; stabilization; cutting stock problem, vehicle routing problem with time windows.

1 Introduction

During the past few decades, the Column Generation (CG) principle has gained considerable popularity for solving various classes of decision problems of practical interest [10]. Although successfully used, this method suffers from

Email addresses: nora.touati@lipn.univ-paris13.fr (N. Touati), lucas.letocart@lipn.univ-paris13.fr (L. Létocart), anass.nagih@univ-metz.fr (A. Nagih).

slow convergence that somewhat limits its efficiency and usability. The column generation scheme offers several possibilities for improvement. There are techniques for accelerating the overall convergence of the CG algorithm, such as bundle [8,23,33], polyhedral [5,14,26], and center [15,18,13,28] stabilization methods, which operate on dual space and aim at stabilizing dual variable behavior. Other methods consist of hybridizing the method with other optimization methods such as the subgradient method [2,4]. In addition, related but different approaches are proposed for accelerating particular CG problems generally based on the resolution of pricing problems. On the one hand, pricing problems can be used to approximate solutions by using heuristics [6,21,29], metaheuristics [7], or relaxations [27], while on another, reoptimization techniques [12,30] can be used. An overview of these methods is presented in [31].

In the primal view of the column generation environment, the oracle outputs a column that enriches the Restricted Master Problem (RMP) description; in dual space, the oracle produces a cutting plane that refines a polyhedral relaxation of the dual function. The efficiency of the column generation method is considerably related to the computed column and dual solution quality at each iteration. Stabilization methods aim at computing "good" dual solutions, close to the best current one. These approaches allow the iterations number to be decreased.

Another more conventional way to reduce the number of iterations in practice is to add a set of columns corresponding to solutions with negative reduced cost (minimization problem case), also including not-optimal solutions. This allows the Master Problem (MP) approximation to be improved, an optimal basis to be characterized more quickly, and hence to decrease the number of iterations. This intensification generally overloads the master problem and generates a large proportion of useless variables, which do not belong in the final optimal basis. The fundamental issue in this work is to study possibilities of taking full advantage of this information set output by the oracle in order to improve the restricted master description without overloading it.

To decrease the restricted master problem size, we generally add only the k-first generated solutions or the k-best marginal cost solutions at each iteration. This paper studies the characteristics and the pertinence of information brought by these columns to the restricted master description, from theoretical and experimental basis. Next, an alternative selection criterion is studied while also looking also at columns structure. In fact, in order to improve the restricted master problem description, it is preferable to select heterogeneous columns [32]. Two different techniques are proposed for computing these particular columns and show theoretically and in experiments, why and how suboptimal heterogeneous columns are preferable over suboptimal best marginal cost ones. In order to appreciate the impact of the proposed approaches, three criteria have been defined:

1. The RMP size must decrease, i.e. the generated columns quality must improve.
2. The overall computing time required for finding an optimal basis for the Master Problem (MP) should decrease.
3. The column generation iteration number must not increase significantly.

Our experimental study focuses on two optimization applications. The vehicle routing problem with time windows based on a time-space network (called Acyclic Vehicle Routing Problem with Time Windows (AVRPTW)) and the one-dimensional cutting stock problem.

This paper is organized as follows. In Section 2 we address the column generation approach on the primal and dual spaces. In Section 3 we discuss different strategies of inserting columns into the master problem at each iteration of column generation: intensification, k-intensification and diversification. We present some diversification procedures, analyze the characteristics of the generated columns and compare diversification with the stabilization principle. Section 4 presents an experimental study of the proposed approaches on the acyclic vehicle routing problem with time windows and on the one-dimensional cutting stock problem. Lastly, in Section 5 we present our conclusions.

2 Column generation

Let us consider the following problem:

$$(IP) \qquad \min cx, \quad Ax \geq a, \quad Bx \geq b, \quad x \in \mathbb{N}^n,$$

where $A \in \mathbb{R}^{m \times n}$, $B \in \mathbb{R}^{p \times n}$, $a \in \mathbb{R}^m$, $b \in \mathbb{R}^p$ and $m, n, p \in \mathbb{N}$. Suppose that the set $X_B = \{x \in \mathbb{N}^n | Bx \geq b\}$ is bounded, it can be expressed as $X_B = \{x_i, i \in I\}$, where I is a finite set of indices. Hence, (IP) can be reformulated as follows:

$$\min \textstyle\sum_{i \in I} (cx^i)\lambda_i, \quad \sum_{i \in I} (Ax^i)\lambda_i \geq a, \;\; \sum_{i \in I} \lambda_i = 1, \;\; \lambda_i \in \mathbb{N}\,, \forall i \in I$$

The continuous relaxation of this new formulation is equivalent to the lagrangian relaxation of constraints $Ax \geq a$ [24]. The Dantzig-Wolfe [9] master problem is obtained:

$$(MP) \quad \min \sum_{j \in J} (cx^j)\lambda_j, \quad \sum_{j \in J} (Ax^j)\lambda_j \geq a, \;\; \sum_{j \in J} \lambda_j = 1, \;\; \lambda_j \geq 0\,, \forall j \in J,$$

where J is the index set of extreme points of $Conv(X_B)$, the convex hull of X_B. As the (MP) number of variables is potentially exponential, we define the master problem (MP^k) restricted to the subset of variables indexed in $J^k \subseteq J$, called Restricted Master Problem. Columns are iteratively added to the RMP until optimality reached. By analogy with the simplex algorithm, we compute at each iteration a column with negative reduced cost, candidate to enter into the basis. The reduced cost $\bar{c}(x)$ of a variable x is given by:

$$\bar{c}(x) = (c - \pi A)x - \pi_0$$

where $\pi \in \mathbb{R}^m_+$ (resp. $\pi_0 \in \mathbb{R}$) is the dual variable vector corresponding to constraints $\sum_{j \in J}(Ax^j)\lambda_j \geq a$ (resp. $\sum_{j\in J} \lambda_j = 1$). The search for a variable with a negative reduced cost is achieved by solving the following problem, called the Pricing Problem (PP):

$$(PP_\pi) \quad \min_{j\in J}\{(c - \pi A)x^j - \pi_0\} \Leftrightarrow \min_{x \in X_B}\{(c - \pi A)x\}$$

Column generation seen in the dual space

Consider the dual problem of (MP):

$$(DM) \quad \max \pi a + \pi_0, \quad \pi(Ax^j) + \pi_0 \leq cx^j, \forall j \in J, \ (\pi, \pi_0) \in (\mathbb{R}^m_+, \mathbb{R}).$$

Let $\theta = \pi a + \pi_0$, the problem (DM) can be reformulated as:

$$\begin{aligned}(DM) \quad &\Leftrightarrow \quad \max \theta, \quad \theta \leq cx^j + \pi(a - Ax^j), \forall j \in J, \ (\theta, \pi) \in (\mathbb{R}, \mathbb{R}^m_+)\\ &\Leftrightarrow \quad \max \theta, \quad \theta \leq \Theta(\pi), \ (\theta, \pi) \in (\mathbb{R}, \mathbb{R}^m_+),\end{aligned}$$

with $\Theta(\pi) = \min_{j\in J}\{cx^j + \pi(a - Ax^j)\}$. The dual problem potentially contains an exponential number of constraints, equal to $|J|$. Kelley's cutting plans method [20] considers a reduced set of these constraints that handle an RMP. Cuts are added at each iteration until the optimum of Θ reached. The problem (DM) can also be formulated as:

$$(DM) \ \Leftrightarrow \ \max \theta, \quad \theta \leq \Theta(\pi^j) + g^j(\pi - \pi^j), \forall j \in J, \ (\theta, \pi) \in (\mathbb{R}, \mathbb{R}^m_+),$$

where g^j is the subgradient of Θ at π^j. From the dual point of view, the pricing problem resolution allows to define a facet of Θ at the considered dual point. CG in the dual space consists in iteratively approaching the function Θ.

3 Solution intensification in column generation

It is well-known that solution intensification which consists in adding several columns to the RMP at each iteration of CG, contributes to decreasing the number of iterations. Unfortunately this can considerably expand the RMP, when the final optimal base contains a very restricted number of the generated columns (see table 2, section 4.1.4). The set of columns of negative reduced cost to be added to the master problem can deeply affect the overall number of generated variables and the computing time required to find an optimal solution. To avoid the rapid and useless increase in RMP size, classically we restrict to the generation of the k-first or k-best solutions (as illustration, see experimental study of section 4.1.4, table 2).

3.1 k-Intensification

Solutions of neighbor reduced cost generally contribute to the same master problem constraints.

Definition 1 *Let be $x, x^p \in \mathbb{N}^n$ and $L(x, x^q) = \{i | x_i.x_i^q = 0, i = 1, ..., n\}$. x^q is p-neighbor compared to x if $|L(x, x^q)| = p$, that is x^q contributes to the same p constraints as x.*

Denote by $l(x^q)$, the length of x^q (number of non-null components of x^q). If $|l(x^q) - |L(x, x^q)|| < e$, for a given small integer e, then solutions x and x^q are too close; the addition of x^q to the restricted master problem not brings significant information to the master problem approximation. So we need to generate fewer columns, but good ones, in order to significantly improve the restricted master approximation. In [32], the author states that a better MP approximation can be obtained when selecting a set of heterogeneous solutions. In the following, we analyze complementary column characteristics and we propose effective ways to release diversification in a CG context.

3.2 Diversification

Let x an optimal solution of the PP. Let x^1 and x^2 two suboptimal solutions with the same cost, respectively $p1$-neighbor and p2-neighbor compared to x such that $p2 < p1$. x associated with x^2 contribute to more constraints than x associated with x^1, that is (x, x^2) contribution to characterize MP domain dominates (x, x^1) contribution. In dual space, the suboptimal cut $C2$ associated with the column x^2 leads to a better dual function approximation with respect to C (cut associated with x) than the suboptimal cut C^1 (associated with x^1).

Remark 1 *Intensification subproblem solution space contains only feasible solutions. Diversification subproblem solution space include feasible solutions that contribute to a maximum number of constraints. Therefor, the diversification subproblem is richer that the intensification one.*

The diversification procedure consists in inserting a set of 0-neighbor columns into the master problem at each column generation iteration. The generation of complementary solutions allows an earlier improvement of the master problem approximation. These procedures can be more efficient on the first iterations to quickly characterize a good approximation and useless on the last ones. Thus, diversification may be applied only on the first iterations.

We present the following two procedures for computing 0-neighbor columns at an iteration of CG.

3.2.1 Diversification by resolution

At each CG iteration and with the same MP dual variables, Diversification by Resolution (CGDR) consists of iteratively computing a 0-neighbor solution compared to all generated columns with optimal reduced cost. Algorithm 1 presents the main steps of one iteration:

Algorithm 1. Schematic iteration of CG with diversification by resolution

```
  $\mathcal{S} \leftarrow$ set of all pricing problem's feasible solutions
  $\mathcal{C} \leftarrow \emptyset$ {set of columns to add to the MP}, next $\leftarrow$ true
  **repeat**
    $x^k \leftarrow$ a solution of minimum cost in $\mathcal{S}$
    **if** $x^k$ has negative cost **then**
      $\mathcal{C} \leftarrow \mathcal{C} \cup \{x^k\}$
      $\mathcal{S} \leftarrow \mathcal{S} \setminus$ {solutions contributing to the same constraints than $x^k$}
      **if** $\mathcal{S} \neq \emptyset$ **then** $k \leftarrow k+1$,
      **else** next $\leftarrow$ false
      **end if**
    **else**
      next $\leftarrow$ false.
    **end if**
  **until** (next = false)
```

3.2.2 Diversification by selection

For subproblems that can be solved by an algorithm giving a set of solutions without over-cost computing time, CG with Diversification by Selection (CGDS) is proposed. It consists in selecting 0-neighbor columns among all the negative reduced cost solutions computed at each iteration of CG. Algorithm 2 presents the main steps of one iteration:

Algorithm 2. Schematic iteration of CG with diversification by selection

```
$\mathcal{C} \leftarrow \emptyset$ {set of columns to add to the MP}
Solve the pricing problem
$X \leftarrow$ all solutions with negative reduced cost
while $X \neq \emptyset$ do
    $\hat{x} \leftarrow$ minimal cost solution in $X$, $\mathcal{C} \leftarrow \mathcal{C} \cup \{\hat{x}\}$
    $X \leftarrow X \setminus$ {solutions contributing to at least one same constraint than $\hat{x}$}
end while
```

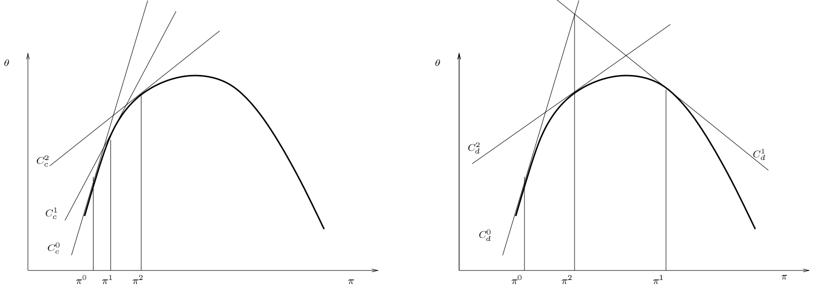
We expected that this technique can lead to the computation of very few columns at each iteration, which include many poor reduced cost ones compared to those generated using the CGDR approach.

3.3 Relationship between diversification and stabilization

CG stabilization methods aim at computing a "good" dual solution at each CG iteration, to decrease the number of iterations. The selection criterion of these dual solutions is based on the computation of a neighborhood around the current best dual solution. This avoids dual solution oscillations. Therefore, the goal of this method is to generate relatively close cuts according to the fixed neighborhood. Figure 1-(a) illustrates the dual function approximation at iteration 3 of CG, with each iteration $i \in \{0, 1, 2\}$ providing a cut C_s^i.

Diversification consists of generating various cuts associated with complementary columns. These cuts have the property of being deeper and allow a global dual function approximation to be effectively characterized. As for figure 1-(a), figure 1-(b) shows the dual function approximation obtained with diversification where we compute at each iteration $i \in \{0, 1, 2\}$, a cut C_d^i corresponding to a complementary column.

When stabilization aims at computing a good dual function local approximation around the best dual solution found, diversification aims at construct a good dual function global approximation.



(a) Stabilisation (b) Diversification

Fig. 1. Stabilization and diversification in column generation

4 Applications

We present in the following two experimental studies on two different classes of problems: the cutting stock problem and the vehicle routing problem. The goal for the same resolution scheme, is to evaluate the impact of inserting or not inserting a set of 0-neighbor solutions. In particular, we show the interest of taking into account although the column structure instead of inserting columns with the best reduced costs.

4.1 Application to the vehicle routing problem with time window

The VRPTW can be described as follows: given a set of costumers, a set of vehicles, and a depot, the VRPTW is to find a set of minimum cost routes, originating and terminating at the depot, such that each costumer is visited by exactly one vehicle to satisfy a specific demand. For each costumer, the service starts during a given time window. A vehicle can wait in case of early arrival, but late arrival is not permitted. In connection with costumer demands, a capacity constraint restricts the load that can be carried by a vehicle.

The master problem can be formulated as a set partitioning problem [1]. For this, let R be the set of all feasible routes, i.e. routes satisfying the time window restrictions. The master problem can be expressed as follows:

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

where c_r is the cost of route r and δ_{ir} is 1 if route r visits node i and 0 if not. The resolution of this formulation by column generation, consists in repeatedly selecting a variable λ_r associated with a route $r \in \mathcal{R}$ with negative reduced cost. This variable is a solution of the pricing problem equivalent to a Shortest Path Problem with Time Windows and Capacity Constraints (SPPTWCC).

The standard approach to solving the SPPTWCC in practice is based on the dynamic programming method and has a pseudo-polynomial time 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. A label correcting algorithm is being considered here, where the nodes are repeatedly treated and their labels extended [11]. Solving the SPPTWCC with a dynamic programming approach is closely related to solving multiobjective shortest path problems, the aim being to generate non-dominated paths (i.e., Pareto optimal paths).

4.1.1 Numerical experiments

Many routing problems in practice are modelled by acyclic networks, particularly when the width of the time windows is lower than the inter-task time of its successor nodes. It is the case in locomotive assignment problems [25], school-bus assignment and aircraft fleet assignment for example [3] where each node in the network represent a time-space state. In this case, a topological order can be established on the nodes. The label correcting algorithm in this case is still not polynomial [22].

4.1.2 Test problems

Two kinds of AVRPTW instances are considered in our tests:

- Some Solomon test instances from which we can extract acyclic networks. The time window's width associated with each node is lower than the maximum of the successor arc durations: (C101 25 (25 costumers), C101 50 (50 costumers), C101 (100 costumers) and C1 2 1 (200 costumers)).
- 40 randomly generated instances. 10 instances are generated for each size: G 100 (100 costumers), G 120 (120 costumers), G 140 (140 costumers) and G 160 (160 costumers), by varying 4 parameters:

(1) The number of adjacent arcs for each vertex is dispersed in the square [degMin, degMax] according to a uniform distribution, where $degMin \in \{5, 7, 8, 10\}$ and $degMax \in \{12, 15, 20, 23, 25\}$.
(2) The time window width is uniformly generated in the square $[5, 20]$.
(3) The demands are uniformly generated in the square $[10, 40]$.
(4) The vehicle's capacity is uniformly generated in the square $[150, 500]$.
(5) The costs are uniformly generated in the square $[10, 25]$. All results reported in this paper for each randomly generated class size are average values over 10 test instances.

4.1.3 Solution intensification

This study began with preliminary experiments on AVRPTW instances solved by the Intensified Column Generation (ICG), where all Pareto optimal columns with negative reduced cost are added to the MP at each iteration. A basic suboptimal column is known as a pricing problem suboptimal solution that belongs to the final optimal basis. Table 1 shows the contribution of suboptimal columns to the final optimal basis.

Table 1
Generation of suboptimal columns

Instance	$C101\_25$	$C101\_50$	$C101$	$C1\_2\_1$	G_{100}	G_{120}	G_{140}	G_{160}
NbCols	246	1 003	2 724	11 461	16 960	19 013	25 259	23 864
%SOCOB	60	87	81	87	97	98	96	98
%GCOB	2,0	0,7	0,5	0,1	0,6	0,2	0,2	0,5

NbCols: total generated columns number.
%SOCOB: pourcentage of basic suboptimal columns in the final optimal basis.
%GCOB: pourcentage of basic optimal columns in NbCols.

These results show that the percentage of basic suboptimal columns in the final optimal basis is higher than 81%, except for the smallest Solomon instance. The early addition of columns to the MP allows an optimal basis to be characterized more quickly, and hence the number of iterations to be decreased. On average, more than 99% of the generated columns do not belong to the final optimal basis. For this purpose, we will study the generated column characteristics in order to propose a selection criterion of good ones. The aim is to reduce the number of generated columns without significantly increasing iteration number for decreasing the global computing time.

4.1.4 k-Intensification

To avoid the rapid and needless increase in the master problem size, we limited ourselves to the generation of the k best solutions, k being a parameter to be determined and expressed in %. The previous case (ICG) is associated with $k = 100$. We call this procedure k% Intensified Column Generation (k%_ICG).

Table 2
Intensified and k% intensified column generation

Solomon's	$C101\_25$				$C101\_50$				$C101$				$C1\_2\_1$			
instances	nbI	nbC	tG	tM	nbI	nbC	tG	tM	nbI	nbC	tG	tM	nbI	nbC	tG	tM
10%_ICG	57	241	11.6”	2.4”	97	895	115,2”	22.6”	220	3655	35.11’	6.61’	309	13159	630,69’	68,34’
20%_ICG	30	306	6.6”	1.5”	59	1027	70,2”	15.9”	101	3433	16.21’	2.82’	232	18721	478,14’	73,93’
50%_ICG	17	517	5.4”	1.6”	31	1644	48.0”	13.6”	66	5329	12.27’	3.02’	159	23679	253,23’	49.01’
ICG	15	785	6.1”	2.2”	25	1957	47.3”	15.2”	42	5614	8.32’	2.13’	109	23622	250,67’	49.67’
Generated	G_{100}				G_{120}				G_{140}				G_{160}			
instances	nbI	nbC	tG	tM	nbI	nbC	tG	tM	nbI	nbC	tG	tM	nbI	nbC	tG	tM
10%_ICG	103	3950	12.76’	3.64’	157	5505	40.61’	9.32’	141	6910	40.61’	14.53’	109	7037	23.84’	8.76’
20%_ICG	69	6384	12.21’	4.20’	102	8505	34.88’	10.12’	92	10881	44.01’	16.73’	73	10470	19.90’	8.54’
50%_ICG	37	11632	10.72’	3.92’	49	14923	35.68’	9.92’	47	18662	41.33’	15.53’	44	17325	20.10’	8.18’
ICG	20	17691	12.34’	2.34’	26	21451	44.51’	2.08’	25	26187	36.09’	2.93’	28	23872	22.99’	0.80’

nbI: total iterations number.
nbC: total generated columns number.
tG: the global resolution time.
tM: cumulated master problems resolution time.

Table 2 shows a comparison between the algorithm performances for different k values. The column number is reduced when k decreases, whereas the iteration number and the resolution time increase in most instances. The addition of columns with good reduced cost is not enough to improve the computing time required to find an optimal basis.

Table 3 shows for the previous procedures the average suboptimal column percentage in the final optimal basis for the same test instances.

	$C101\_25$	$C101\_50$	$C101$	$C1\_2\_1$	G_{100}	G_{120}	G_{140}	G_{160}
10%_ICG	75	85	77	89	85	83	87	88
20%_ICG	25	100	84	88	89	93	92	96
50%_ICG	75	100	92	89	96	96	95	97
ICG	60	100	93	93	97	98	97	98

Table 3
Suboptimal columns proportion in the final optimal basis (in %)

On average, this percentage increases when k increases, so suboptimal solutions with good reduced cost contribute less to the final optimal basis than those with a worse reduced cost. These experimental results show that at each iteration of k%_ICG (k = 10, 20, 50) procedures, suboptimal columns generated have generally slight deviations compared to the optimal solution obtained, i.e. they cover almost the same nodes.

4.1.5 Diversification

Diversification by resolution consists in repeatedly solving the PP while removing from the network all the nodes covered by the optimal solution of negative reduced cost that was obtained, with the next PP being solved on the partial subgraph that is induced. Diversification by selection consists in selecting from all the Pareto optimal solutions of negative reduced cost obtained the best 0-neighbor ones. Based on ICG, CGDR and CGDS consist in respectively applying diversification by resolution and diversification by selection 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 and ϵ a given small real. Table 4 presents the results.

Table 4
Diversification in column generation

Solomon's	$C101\_25$				$C101\_50$				$C101$				$C1\_2\_1$			
instances	nbI	nbC	tG	tM	nbI	nbC	tG	tM	nbI	nbC	tG	tM	nbI	nbC	tG	tM
20%_ICG	30	306	6.6"	1.5"	59	1027	1.17'	15.9"	101	3433	16.21'	169.2"	232	18721	478,14'	73,93'
50%_ICG	17	517	5.4"	1.6"	31	1644	0.80'	13.6"	66	5329	12.27'	181.2"	159	23679	253,23'	49.01'
ICG	15	785	6.1"	2.2"	25	1957	0,78'	15.2"	42	5614	8.32'	127,8"	109	23622	250,67'	49,67'
CGDS	46	344	10.6"	0.06"	56	1069	1.09'	0.3"	95	3726	15.63'	3.3"	136	20101	219,05'	0,70'
CGDR	42	343	10.2"	0.06"	60	1221	1.27'	0.4"	73	3293	9.83'	1.5"	126	18285	221,10'	0,57'
Generated	G_{100}				G_{120}				G_{140}				G_{160}			
instances	nbI	nbC	tG	tM	nbI	nbC	tG	tM	nbI	nbC	tG	tM	nbI	nbC	tG	tM
20%_ICG	69	6384	12.21'	252.0"	102	8505	34.88'	607.2"	92	10881	44.01'	1003.8"	73	10470	19.90'	512.5"
50%_ICG	37	11632	10.72'	235.2"	49	14923	35.68'	595.2"	47	18662	41.33'	931.8"	44	17325	20.10'	490.8"
ICG	20	17691	12.34'	142.2"	26	21451	44.51'	124.8"	25	26187	36.09'	175.5"	28	23872	22.99'	48.3"
CGDS	48	3573	7.34'	1.8"	51	5676	22.29'	4.5"	52	7207	19.35'	6.0"	57	4822	9.28'	3.3"
CGDR	40	1545	7.21'	1.0"	42	2684	22.32'	2.2"	45	2552	19.89'	2.2"	49	1885	10.08'	1.3"

nbI: total iterations number.
nbC: total generated columns number.
tG: the global resolution time.
tM: cumulated master problems resolution time.
$\varepsilon = 0.001$.

As expected, CGDS allows the total columns number to be decreased; we generate on average only 43% of what was generated by the ICG. The number of

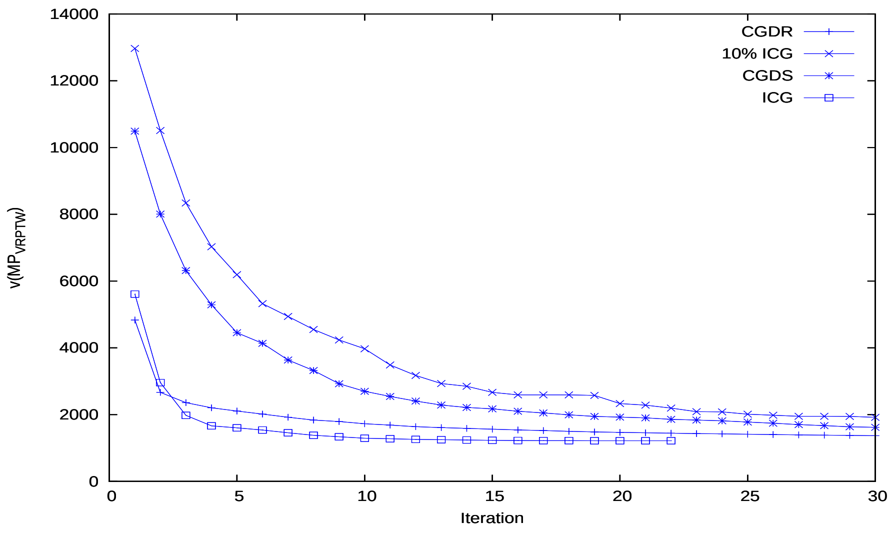


Fig. 2. Master problems evolution values on the 30^{th} first iterations

columns generated by CGDS at each iteration is very small compared to ICG, but their costs are bad (relatively high). The intensification of complementary columns with better costs (CGDR) allows the computed columns number to be decreased; we generate on average 41% of what was generated by ICG. Master problems are then smaller and their resolution easier. This decreases the global resolution time of ICG procedures by 36% with CGDS and 47% with CGDR on average.

The dominant procedure between CGDR and CGDS is that which achieve out a compromise between the gain on time obtained from the MP resolution and the additional cost of pricing problem resolution. With a reduced number of columns, CGDR built a best approximation of the pricing problem's convex hull. Figure 2 compares the master problem's values ($v(MP_{VRPTW})$) on the 30 first iterations when we add to the MP all the columns of negative reduced cost (ICG), the columns with best reduced costs (10%_ ICG) and the complementary columns (CGDS and CGDR). For example, on the two first iterations, the approximation obtained by the CGDR with 76 columns is better than the ICG one with 12037 columns. Diversification allows the generated column number to be decreased without significantly increasing the iteration number and to reduce the global computing time of the larger test instances where size of the master problem is great, which is the case in our experimentations when $n \geq 100$, where n is the number of nodes (Table 4).

4.2 Application to the one-dimensional cutting stock problem

The One-dimensional Cutting stock Problem (1D-CSP) consists in minimizing the number of rolls of paper (each with length L) needed to produce b_i rolls with length l_i for $l_i = 1, 2, ..., m$, where $0 <= l_i <= L$ for each i.

The master problem can be formulated as a set covering problem [16,17] corresponding to demand satisfaction. Let D be the set of all feasible cutting patterns with respect to the length L and let a_{ij}, $i = 1, ..., m$ and $j \in D$ be the number of pieces of length l_i cut in one roll of configuration j. The master problem can be expressed as follows:

$$\begin{aligned} \min \quad & \textstyle\sum_{i=1}^{n} x_i \\ & \textstyle\sum_{i=1}^{n} a_{ij} x_i \geq b_j, \quad j = 1, ..., m \\ & x_i \geq 0, \text{ integer}, \quad i = 1, ..., n \end{aligned}$$

where x_i denotes the number of rolls cut according to the i-th pattern. With the possibility that $|D|$ is very large, it can be very difficult to solve the master problem directly. We consider a restricted master problem defined by a few patterns and generate new ones until an optimal basis is obtained.

The pricing problem is an integer knapsack problem formulated as follows:

$$\begin{aligned} \max \quad & \textstyle\sum_{j=1}^{m} u_j a_i \\ & \textstyle\sum_{j=1}^{m} l_j a_j \leq L, \\ & a_i \geq 0, \text{ integer}, \quad i = 1, ..., n \end{aligned}$$

where u_j, $j = 1, ..., m$, is the dual variable associated with constraints $\sum_{i=1}^{n} a_{ij} x_i \geq b_j$.

4.2.1 Intensification and diversification

Generally, the 1D-CSP column generation resolution consists in inserting one column of negative reduced cost at each iteration. Most of the improving methods are based on the pricing problem's resolution acceleration. Through these experimentations we attempt to evaluate the impact of intensification and diversification on column generation performances for the resolution of the 1D-CSP. Three CG procedures are being compared:

- Classical CG (CCG) where one column is added to the RMP at each iteration.
- Intensified CG (ICG) where many solutions are added to the RMP at each iteration. To generate many solutions at each iteration, we repeatedly solve the pricing problem while avoiding the computation of the same cost solutions.

- CG with Diversification by Resolution (CGDR), where 0-neighbor columns are added to the RMP at each iteration. To generate 0-neighbor solutions at each iteration, we repeatedly solve the pricing problem while fixing the objective function coefficient value associated with each non-null component of the current solution at a great value. 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 and ϵ a given small real.

A computational experiment was conducted on random problem instances generated by CUTGEN1 [19]. The 1800 instances of 18 classes are solved here. The restricted master problems and the subproblems are solved by CPLEX 10.

Table 5
Diversification in column generation

CUTGEN1	Type 01			Type 02			Type 03			Type 04			Type 05			Type 06		
instances	nbI	nbC	tG	nbI	nbC	tG	nbI	nbC	tG	nbI	nbC	tG	nbI	nbC	tG	nbI	nbC	tG
Av(CCG)	22	22	0.58'	22	22	0.59'	39	39	1.17'	39	39	1.16'	68	68	2.49	68	68	2.82'
Av(ICG)	18	92	1.00'	18	92	0.99'	30	163	2.08'	30	163	2.16'	51	328	5.64'	51	328	5.51'
Av(CGDR)	13	27	0.52'	13	27	0.51'	19	50	1.00'	19	50	0.97'	28	89	2.11'	28	90	2.12'
CUTGEN1	Type 7			Type 8			Type 9			Type 10			Type 11			Type 12		
instances	nbI	nbC	tG	nbI	nbC	tG	nbI	nbC	tG	nbI	nbC	tG	nbI	nbC	tG	nbI	nbC	tG
Av(CCG)	12	12	0.20'	12	12	0.21'	30	30	0.75'	30	30	0.73'	70	70	3.07'	70	70	3.08'
Av(ICG)	8	34	0.31'	8	34	0.30'	22	64	0.86'	22	64	0.85'	54	135	3.35'	54	135	3.37'
Av(CGDR)	5	13	0.16'	5	13	0.15'	11	31	0.51'	11	32	0.51'	28	73	2.16'	28	72	2.16'
CUTGEN1	Type 13			Type 14			Type 15			Type 16			Type 17			Type 18		
instances	nbI	nbC	tG	nbI	nbC	tG	nbI	nbC	tG	nbI	nbC	tG	nbI	nbC	tG	nbI	nbC	tG
Av(CCG)	10	10	0.14'	10	10	0.14'	26	26	0.51'	26	26	0.51'	69	69	1.79'	69	69	1.82'
Av(ICG)	8	11	0.13'	8	11	0.13'	22	28	0.43'	22	28	0.45'	57	69	1.54'	56	69	1.52'
Av(CGDR)	3	10	0.10'	3	10	0.10'	6	24	0.26'	6	24	0.27'	13	55	0.77'	13	55	0.74'

nbI: total iterations number.
nbC: total generated columns number.
tG: the global resolution time.
$\varepsilon = 0.001$.

As can be seen from Table 5, solutions intensification decrease the iteration number but increase the number of generated columns compared to the CCG. The global resolution time is decreased for Type i, $i = 13, ..., 18$ instances. Diversification (CGDR) permits to decrease the iteration number compared to the ICG, therefor, optimal solutions are obtained more quickly with efficient characterization of the MP description. This decrease the total number of generated columns. The CGDR procedure decreases the CCG resolution time by 29% and the ICG resolution time by 44%.

5 Conclusion

Column generation stabilization methods aim at computing good dual solutions at each iteration, thus decreasing the iteration number. Dual solution quality affects that of the primal solutions, so the generated columns quality is important for improving column generation performance. We focused on the intensified column generation where a set of columns is added to the master problem at each iteration. We studied some characteristics of the generated columns and some properties of good ones to avoid needless columns and decreasing the master problem size. Experimental results on Solomon and randomly generated instances of the acyclic vehicle routing problem with time windows indicate that the addition of the k% pricing problem best solutions (k=10, 20, 50) to the master problem at each iteration increase the iterations number and column generation resolution time.

Instead of interesting to the suboptimal columns reduced cost, we interest to their structure. It is known that complementary columns efficiently improve the restricted master problem description. We proposed two different ways to compute these particular columns: diversification by selection and diversification by resolution. We presented in this paper the first study on the impact of diversification on the intensified column generation performance, where two problems are considered: the acyclic vehicle routing problem with time windows and the one-dimensional cutting stock problem.

Experimental results on our test instances show that the generation of complementary columns allows the total generated column number and master problem resolution time to be significantly decreased. Diversification for the acyclic vehicle routing problem more efficiently improves the restricted master problem description than k%-Intensification. The diversification methods proposed for this problem dominate the intensified column generation for the largest instances, despite the additional difficulty imposed on pricing problems resolution. The efficiency of diversification (by resolution) is also shown in the one-dimensional cutting stock problem.

We distinguished here a correlation between diversification and stabilization principles, which were to be compared on an experimental level, between diversification and stabilization column generation schemes. We analyzed in this paper the impact of diversification in a column generation algorithm for the computation of a discrete problem Lagrangian relaxation bound. It will be interesting to study this impact on the overall branch-and-price scheme where column generation with diversification algorithm is used to solve the problem at each node of the branch-and-bound tree.

References

[1] Y. Agarval, K. Mathur and H. M. Salkin, A set-partitioning-based exact algorithm for the Vehicle Routing Problem, Networks, 19(7):731-749, 1989.

[2] M. Akker, H. Hoogeveen and S. de Velde, Combining column generation and lagrangean relaxation to solve a single-machine common due date problem,

INFORMS Journal on Computing, 14(1):37-51, 2002.

[3] M.O. Ball, T.L. Magnanti, C.L. Monma, G.L. Nemhauser, Network Routing, Handbooks in Operations Research and Management Science, volume 8, 1995.

[4] F. Barahona and D. Jensen, Plant location with minimum inventor, Mathematical Programming 83(1):101-112, 1998.

[5] H. Ben-Amor, J. Desrosiers and A. Frangioni, Stabilization in column generation, Les Cahiers du Gerad, HEC Montréal, G-2004-62, 2004.

[6] O. Braysy and M. Gendreau, Vehicle routing problem with time windows, Part I: Route construction and local search algorithms, Transportation Science, 39(1):104-118, 2005.

[7] O. Braysy and M. Gendreau, Vehicle Routing Problem with Time Windows, Part II: Metaheuristics, Transportation Science, 39(1):119-139, 2005.

[8] O. Briant, C. Lemaréchal, Ph. Meurdesoif, S. Michel, N. Perrot and F. Vanderbeck, Comparison of bundle and classical column generation, Mathematical Programming, 113(2):299-344, 2006.

[9] G.B. Dantzig and P. Wolfe, Decomposition principle for linear programs, Operations Research, 8:101-111, 1960.

[10] G. Desaulniers, J. Desrosiers and M. M. Solomon, Column generation, Springer, New York, 2005.

[11] M. Desrochers, An algorithm for the Shortest Path Problem with Resource Constraints. Technical Report G-88-27, GERAD, 1988.

[12] M. Desrochers and F. Soumis, A reoptimization algorithm for the shortest path problem with time windows, European Journal of Operational Research, 35:242-254, 1988.

[13] O. Du Merle, J.L. Goffin and J.P. Vial, On Improvements to the Analytic Center Cutting Plane Method, Computational Optimization and Applications, 11(1):37-52, 2004.

[14] O. Du Merle, D. Villeneuve, J. Desrosiers and P. Hansen, Stabilized column generation, Discrete Mathematics, 194:229-237, 1999.

[15] J. Elzinga and T.G. Moore, A central cutting plane algorithm for the convex programming problem, Mathematical Programming, 27:134-145, 1973.

[16] P.C. Gilmore and R.E. Gomory, A linear programming approach to the cutting-stock problem, Operations Research, 9(6):849-859, 1961.

[17] P.C. Gilmore and R.E. Gomory, A linear programming approach to the cutting-stock problem-Part II. Operations Research 11(6):863-888, 1963.

[18] J.L. Goffin and J.P. Vial, Cutting planes and column generation techniques with the projective algorithm, Journal of Optimization Theory and Applications, 65:409-429, 1990.

[19] T. Gau and G. Wäscher, CUTGEN1: a problem generator for the standard onedimensional cutting stock problem, European Journal of Operational Research, 84:572-579, 1995.

[20] J.E. Kelley, The cutting-plane method for solving convex programs, SIAM Journal on Optimization, 8:703-712, 1960.

[21] G. Laporte, The vehicle routing problem: An overview of exact and approximate algorithms, European Journal of Operational Research, 59:345-358, 1992.

[22] O. Laval, A. Nagih and S.Toulouse, Rapport de recherche sur le problème du plus court chemin contraint, Technical Report 2006-05, LIPN - CNRS UMR 7030 - Université Paris 13, 2006.

[23] C. Lemaréchal, An algorithm for minimizing convex functions, Information Processing, 74:552-556, 1974.

[24] C. Lemaréchal, The omnipresence of Lagrange, Annals of Operations Research, 153(1):9-27, 2007.

[25] N. Marcos, Modélisation et Optimisation de la Gestion du Matériel Roulant à la SNCF, Thesis, Laboratoire d'Informatique de Paris-Nord (LIPN) - Université Paris 13 - Villetaneuse de Paris, 2006.

[26] R.M. Marsten, W. Hogan and J. Blankenship, The Boxstep method for large-scale optimization, Operations Research, 23:389-405, 1975.

[27] A. Nagih and F. Soumis, Nodal aggregation of resource constraints in a shortest path problem, European Journal of Operational Research, 172:500-514, 2006.

[28] A. Ouorou, A proximal cutting plane method using Chebychev center for nonsmooth convex optimization, Mathematical Programming, to appear, 2008.

[29] S. Ropke and D. Pisinger, A unified heuristic for a large class of vehicle routing problems with backhauls, European Journal of Operational Research, 171:750-775, 2006.

[30] B. Thiongane, A. Nagih and G. Plateau, Adapted step size algorithm for a 0-1 biknapsack lagrangean dual, Annals of Operations Research, 139(1):253-373, 2004.

[31] N. Touati, L. Létocart and A. Nagih, Sur l'accélération de la convergence de la génération de colonnes, Technical report LIPN-2006-4, University Paris 13, 2006.

[32] F. Vanderbeck, Decomposition and Column Generation for Integer Programs, PhD thesis, UCL 1994.

[33] P. Wolfe, A method of conjugate subgradients for minimizing nondifferentiable functions, Nondifferentiable Optimization, Mathematical Programming Study, 3:145-173, 1975.