Improving the Cooperation Between the Master Problem and the Subproblem in Constraint Programming Based Column Generation

Bernard Gendron[1,2], Hocine Lebbah[2,3], and Gilles Pesant[2,3]

[1] Département d'informatique, et de recherche opérationnelle, Université de Montréal, C.P. 6128, succ. Centre-ville, Montreal, Quebec H3C 3J7
bernard@crt.umontreal.ca
[2] Centre de recherche sur les transports, Université de Montréal, C.P. 6128, succ. Centre-ville, Montreal, Quebec H3C 3J7
[3] Département de génie informatique, École Polytechnique de Montréal, C.P. 6079, succ. Centre-ville, Montréal, Québec H3C 3A7

Abstract. Constraint programming (CP) based column generation uses CP to solve the pricing subproblem. We consider a set partitioning formulation with a huge number of variables, each of which can be generated by solving a CP subproblem. We propose two customized search strategies to solve the CP subproblem, which aim to improve the coordination between the master problem and the subproblem. Specifically, these two strategies attempt to generate more promising columns for the master problem in order to counter the effect of slow convergence and the difficulty of reaching integer solutions. The first strategy uses the dual variables to direct the search towards columns that drive the relaxed master problem faster to optimality. The second strategy exploits the structure of the constraints in the master problem to generate columns that help to reach integer solutions more quickly. We use a physician scheduling problem to test the strategies.

1 Introduction

Constraint programming (CP) based column generation uses CP to solve the pricing subproblem in a column generation algorithm. First introduced by Junker et al. [10], it has been subsequently used for various applications such as airline crew scheduling [6, 17] and vehicle routing [15]. In these applications, the master problem corresponds to a set partitioning model and the pricing subproblem is formulated as a constrained shortest path problem. Although dynamic programming is generally used to solve constrained shortest path problems, some complex constraints can be difficult to handle within this framework. In this context, CP allows more flexibility and can extend the scope of applicability of column generation algorithms. It is also worthwhile to note that not all practical applications give rise to constrained shortest path subproblems. For example, in a constrained cutting-stock problem, the pricing subproblem is a constrained knapsack problem. Again, in this context, CP can be useful since the subproblems can include complex constraints in addition to the knapsack structure [7].

Column generation for linear programming (LP) [4] and integer programming (IP) [8, 9] dates back to the 60s. During the last 20 years or so, it has been used extensively to solve IP problems with a huge number of variables. In this context, the LP relaxation is solved by column generation and integer solutions are obtained by branching. When column generation is repeated at each node of the branch-and-bound tree, we obtain the so-called branch-and-price algorithm [1, 5, 18, 19]. As pointed out by many authors, the main difficulty when solving these huge IPs by column generation is to achieve the right balance between two different objectives: 1) try to solve the LP relaxation of the master problem, and 2) try to obtain an integer solution to the master problem. In branch-and-price algorithms, the second objective is attained by clever branching rules, while the first one is achieved through the coordination between the master problem and the pricing subproblem provided by the dual variables associated to the constraints of the LP relaxation.

The goal of this paper is to show that, in the context of CP based column generation, these objectives can also be reached by selecting an appropriate branching scheme in the pricing subproblem, e.g., by devising customized selection strategies in a standard CP based tree search algorithm. These selection strategies will guide the search towards "good" columns and thus help the master problem to reach LP relaxation optimality, as well as integrality. Note that using such customized selection strategies in solving the pricing subproblem makes the CP based column generation framework even more attractive.

The paper is organized as follows. First, we present the formulation of the master problem: we consider a set partitioning formulation with different restrictions on the subsets. Since many applications found in the literature fall into this category (many others are also set partitioning models, but with the same restrictions on the subsets), we will use this framework to develop our selection strategies. The next section discusses how to model the pricing subproblem and describes two selection strategies: the first one uses the dual variables to direct the search towards columns that drive the LP master problem to optimality; the second strategy exploits the structure of the constraints in the master problem to generate columns that help to reach integer solutions. We illustrate these strategies on a difficult personnel scheduling problem, dealing with the planning of work schedules for physicians in the emergency room of a major hospital. Section 4 presents the particular CP model used in this case. In Section 5, we present computational results that illustrate the behavior of the search strategies in the context of a price-and-branch implementation (column generation to solve the LP, followed by branch-and-bound on the limited set of variables obtained after column generation). We conclude by summarizing our results and by discussing extensions to our work.

2 The Master Problem

Suppose we have a set of shifts K to assign to a set of employees I. Each shift must be assigned to exactly one employee. There are several constraints that limit the number of feasible schedules for each employee (see Section 4 for an illustrative example). To model this problem, we introduce a set P_i of feasible schedules for each employee $i \in I$. Each feasible schedule $p \in P_i$ is made of a number of shifts; we let δ_k^p equal 1, if shift k belongs to schedule p, and 0 otherwise. Assuming that there is a cost c_i^p for assigning schedule p to employee i, we obtain one of the most common forms of huge IP models amenable to solution by column generation, the set partitioning formulation with different restrictions on the subsets [1]:

$$\min \sum_{i \in I} \sum_{p \in P_i} c_i^p y_i^p \tag{1}$$

$$\sum_{i \in I} \sum_{p \in P_i} \delta_k^p y_i^p = 1, \quad k \in K, \tag{2}$$

$$\sum_{p \in P_i} y_i^p = 1, \quad i \in I, \tag{3}$$

$$y_i^p \in \{0,1\}, \quad i \in I, p \in P_i. \tag{4}$$

The partitioning constraints, (2), simply state that each shift must be assigned to exactly one employee. Constraints (3), often called convexity constraints, assure that each employee gets one schedule, when combined with the integrality constraints (4). This model is the column generation formulation of a classical scheduling problem, where each shift must be assigned to exactly one employee and there are several constraints for each employee. In some applications, multiple employees can be assigned to the same shift, in which case the partitioning constraints (2) are simply rewritten with a right-hand side corresponding to the number of employees assigned to that shift. This model arises in many applications, such as airline crew scheduling [6, 17] and the classical generalized assignment problem [16] (the pricing subproblem in this case is a collection of knapsack problems). When the pricing subproblems are identical for all employees, we can aggregate the y_i^p variables and replace them by $y^p = \sum_{i \in I} y_i^p$. The convexity constraints are then often removed because it is common in many applications to have $|I|$ not fixed. Examples of this type of master model arise in vehicle routing [15], as well as in the classical cutting-stock problem [8, 9].

Since the number of variables in this set partitioning formulation is huge, they are generated dynamically by solving the LP relaxation of a restricted model (with only a subset of the variables) and by collecting the values of the dual variables associated to the constraints of the problem. These dual values are then transfered to the pricing subproblem that will try to find at least one variable with negative reduced cost (such variables have not yet been generated, since at optimality of the restricted LP relaxation, all variables have non negative reduced costs). In our case, if we denote by λ_k and μ_i the dual variables corresponding to constraints (2) and (3), respectively, the reduced cost for variable y_i^p is given by

$$\overline{c}_i^p = c_i^p - \sum_{k \in K} \delta_k^p \lambda_k - \mu_i. \tag{5}$$

The pricing subproblem decomposes into one problem for each employee, defined by the constraints corresponding to that employee. The objective of that employee subproblem is to find the solution with the least reduced cost. In practice, we do not want to solve this subproblem optimally; it is generally enough to identify a small number of negative reduced cost solutions. Once these variables have been identified, they are added to the restricted LP relaxation and we proceed with another iteration, until the LP relaxation is solved (to prove optimality, we need an exact algorithm for solving the pricing subproblem) or a maximum number of iterations has been reached. Typically, the LP relaxation of such huge set partitioning formulations are very difficult to solve as they exhibit degeneracy and slow convergence (see [11] and [14] for recent contributions on improving the solution of the LP relaxation in column generation algorithms).

Once the LP relaxation is solved (in an exact or heuristic way) an integer solution can be found by branching. One alternative is to perform branch-and-bound using the set of columns obtained after solving the LP relaxation, but this is a heuristic method, as some non generated columns might appear in an optimal solution. If the column generation scheme is repeated at each node of the search tree, we obtain a branch-and-price algorithm, which is an exact method, provided the pricing subproblem can be solved to optimality.

3 Search Strategies for the Pricing Subproblem

Recall that the pricing subproblem decomposes by employee. In each employee subproblem, we will assume that the schedule can be decomposed by day because no more than one shift can be assigned to the employee on any given day. This allows us to define one variable x_j for each day $j \in J$ in the employee subproblem; thus we have a vector of variables $X = (x_1, x_2, ..., x_n)$, where n is the number of days in the planning horizon. The domain of each variable x_j, $D(x_j)$, is defined as the set of shifts required on day j, plus a dummy shift that represents the outcome that the employee does not work on day j. One can take this dummy shift into account in the master problem by adding partitioning constraints for each day corresponding to the dummy shift, with a right-hand side equal to (number of employees - number of shifts required on that day). In this way, dual values associated to these dummy shift constraints can be taken into account when solving the pricing subproblem.

Another alternative for modeling the employee subproblem is to use a set variable representing the shifts that can be assigned over the whole planning horizon [6]. In this setting, a constraint such as "no more than one shift per day" would be modeled using a *shift graph*, where each node corresponds to a shift on a given day and each arc connects two possible consecutive shifts; thus, in this graph, there would be no arc connecting two shifts on the same day. With our formulation, these constraints are implicitly taken into account. Moreover, we will use them to define global constraints specialized to the physician scheduling problem that we consider in our study (see Section 4). Note that adapting our search strategies to the model with set variables is a straightforward task.

Often, the dominant terms in the reduced costs, (5), are the dual values corresponding to the λ_k variables (since feasibility in the master is an issue). Moreover, the costs associated to the assignment of a particular shift do not differ significantly by employee (this is the case in the application presented in Section 4). In this situation, by solving the employee subproblems independently with the same search strategy, we would end up with many schedules that are similar from one employee to another. This implies that reaching an integer solution that satisfies the partitioning constraints (2) is a difficult issue because several columns for many employees would share the same shifts.

The first search strategy, based on the values of the dual variables, attempts to drive the search towards solutions with negative reduced costs. The objective of this *Dual strategy* is thus to speed up the resolution of the LP relaxation of the master problem. A similar strategy, called Lowest Reduced First, has been proposed by Fahle et al. [6]. In this strategy, the shift with the smallest reduced cost over the whole planning horizon is selected and assigned to the employee. In our approach, we aim to introduce more variability in the selection strategy from one employee to another to avoid generating similar schedules. We achieve this objective by scanning the days and, for each day, by selecting the shift with the lth largest dual value λ_k (which is the dominant term in the reduced cost), where l is randomly chosen, with a bias towards the largest values. By choosing different seed values for the random number generator, we obtain variability in the schedules generated for different employees.

The second strategy tries to speed up the search for integer solutions in the branch-and-bound (-and-price) phase; we call it the *Master strategy*, since the idea is to take into account the partitioning constraints (2) of the master problem in the selection process. We simply store N_{sj}, the number of times each shift s on any given day j has been assigned to another employee when solving the current pricing subproblem. When solving an employee subproblem, we scan the days, and for each day j, we choose the first shift s (in arbitrary order) for which N_{sj} is less than the right-hand side of the partitioning constraint corresponding to shift s (note that this definition allows for right-hand side values larger than 1). If there is no such shift, we choose a shift arbitrarily. In addition, to avoid generating similar schedules from one column generation iteration to the next, we change the order for solving the employee subproblems. Note that, instead of choosing the shifts in arbitrary order, we could have biased the selection towards the shifts with the largest dual values, as in the dual strategy. However, we first wanted to examine the effects of each strategy independently.

We will present computational results of these two strategies on a physician scheduling problem, to be described in the next section. Before examining the particular constraints of that problem, we comment on how we have modeled the negative reduced cost constraint. As in [6], we have used a simple version where the constraint is used only for pruning and not for domain reduction. As shown by these authors, this version of the reduced cost constraint is clearly inferior, especially for large-scale instances, to the shortest path constraint they developed, which performs domain reduction in an efficient way. Since we focus on the search strategies in the pricing subproblems and their impact on solving the master problem, we expect that our conclusions will hold as well if we use shortest path constraints instead of the simple version of the reduced cost constraints.

4 Illustrative Example: A Physician Scheduling Problem

As illustrative example of our approach, we use a simplified version of the physician scheduling problem described in [2] (a similar problem, also modeled with CP, can be found in [3]). In this problem, a group of physicians must be assigned to a predefined set of shifts in order to satisfy the requirements of an emergency room of a major hospital. The schedules are planned for the next n days; typically, the planning horizon varies between two weeks and six months.

Several types of rules must be satisfied in order to obtain acceptable working schedules. First, there are a few compulsory rules, e.g., rules that must absolutely be enforced. Demand rules are the most basic in this category. They define how many physicians should work at different periods of a day and which responsibilities are attached to particular shifts. Each day is divided into three periods of eight hours: day, evening and night. Three physicians (two on weekends or holidays) work during day and evening shifts, including one exclusively in charge of traumas ("heavy" emergencies). "Trauma" shifts are considered heavier than "regular" shifts (which mostly involve the treatment of "light" cases and patients in stabilized condition). At night, there is only one night shift, the physician assuming the responsibilities of trauma and regular shifts. Three days per week, one physician works a four-hour shift, the "follow-up" shift, when he receives by appointment patients that have recently been treated at the emergency room. Other compulsory rules include: vacations, days-off, or particular shifts requested by the physicians, and the basic ergonomic rule: "there must be at least 16 hours between the end of one shift and the beginning of another one".

The demand rules are implicitly taken into account in the definition of our variables. The other rules dealing with preassigned shifts are easily modeled, as well as the basic ergonomic rule, which is formulated as follows:

$$x_j = s \Rightarrow x_{j+1} \neq t, \quad j \in \{1, 2, ..., n-1\},\, s \in S_j,\, t \in F_{sj}, \tag{6}$$

where S_j is the set of required shifts on day j and F_{sj} is the set of forbidden shifts once shift s is assigned on day j (all forbidden shifts are located on day $j+1$). Another way to model this constraint is by using the shift graph described in Section 3.

In addition to these basic constraints, the physicians have their own requirements regarding the number of consecutive night shifts they accept to work: some prefer to work three consecutive nights, while others do not want to work more than one night at a time, and some others accept any number of consecutive nights, as soon as it never exceeds three. These specific requirements can be easily modeled using the **stretch** constraint [12]. If we denote by $S = \{1, 2, \ldots, m\}$ the set of shifts that can be assigned on any given day (e.g., $S = \cup_{j \in J} S_j$),

$$\mathbf{stretch}(X, S, L^-, L^+) \tag{7}$$

ensures that, in any instanciation of the variables $X = (x_1, x_2, \ldots, x_n)$, the length of any sequence of consecutive days assigned to shift $s \in S$ is between l_s^- and l_s^+, where $L^- = (l_1^-, l_2^-, \ldots, l_m^-)$ and $L^+ = (l_1^+, l_2^+, \ldots, l_m^+)$. The stretch constraint also serves to limit to four (in some cases, five) the number of consecutive shifts of any type.

In addition, there are constraints on the minimum and maximum number of hours each physician can work every week. These constraints are modeled using the two global constraints **distribute** [13] and **ext**. The constraint $\mathbf{distribute}(C, S, X)$ ensures that each value $s \in S$ appears exactly c_s times in X, where $C = (c_1, c_2, \ldots, c_m)$. The constraint $\mathbf{ext}(M, \Gamma)$ lists all the admissible d-tuples of variable $M = (m_1, m_2, \ldots, m_d)$. Let

- H, an m-dimensional vector such that H_s equals the number of hours worked during shift s (there are only three possible values: 0, for the dummy shift, 4, for the follow-up shift, and 8, for all other shifts);
- $Y = (y_1, y_2, \ldots, y_n)$, an n-dimensional vector of variables defined as $y_j = H_{x_j}$;
- $M = (m_0, m_4, m_8)$ a 3-dimensional vector of variables representing the number of times 0-hour, 4-hour and 8-hour shifts appear in Y.

The constraint on the minimum ($minH$) and maximum ($maxH$) number of working hours every week can then be written as:

$$Y = H_X \wedge \mathbf{distribute}(M, \{0, 4, 8\}, Y) \wedge \mathbf{ext}(M, \Gamma), \tag{8}$$

where the set of admissible triples Γ is given by $\Gamma = \{(m_0, m_4, m_8) \in [0, \overline{m_0}] \times [0, \overline{m_4}] \times [0, \overline{m_8}] \mid minH \leq 4m_4 + 8m_8 \leq maxH,\ m_0 + m_4 + m_8 = 7\}$, with $\overline{m_4} = \lfloor maxH/4 \rfloor$, $\overline{m_8} = \lfloor maxH/8 \rfloor$ and $\overline{m_0} = 7 - (\overline{m_4} + \overline{m_8})$.

Many other rules exist in the application described in [2], but our simplified version contains only the constraints described so far. This choice allows to capture enough of the complexity of the problem, and to illustrate as well the flexibility of the modeling approach.

The objective function is defined so as to ensure that different types of shifts are fairly distributed among physicians. In our simplified version, we try to achieve a fair distribution of two types of antagonist shifts: days versus evenings, and regular versus trauma. Positive and negative deviations with respect to an ideal ratio are penalized, thus defining the cost of a schedule over the whole planning horizon.

5 Computational Results

The objective of our computational experiments is to compare the two search strategies, Dual and Master, when embedded within a simple column generation algorithm that proceeds in three phases:

– *Initialization*: During this phase, an initial set of columns is generated. For this purpose, we use a CP engine to solve a global problem containing all the constraints of the subproblem, as well as the partitioning constraints; this method ensures that at least one globally feasible set of schedules is generated (a similar approach is presented in [17]).
– *Column generation*: This phase is the CP based column generation method, with either the Dual or the Master search strategy being used to solve the subproblem; the search is stopped for each employee subproblem when one solution with negative reduced cost is obtained. We stop this phase after a predetermined number of iterations (40 in our tests).
– *Branch-and-bound*: We invoke a branch-and-bound algorithm that solves the formulation obtained at the end of the column generation phase. This allows to evaluate if the columns generated during the column generation phase are sufficient to improve the initial integer solution.

The overall method is programmed with ILOG Concert (version 1.1). ILOG Solver (version 5.1) is used for solving the subproblems and the global problem in the initialization phase. ILOG CPLEX (version 7.1) solves the restricted LP relaxations in the column generation phase, as well as the restricted IP model in the branch-and-bound phase. The default parameters are used for all software packages.

We tested the approach on an instance of the physician scheduling problem with 18 physicians over a period of two weeks. Table 1 compares the results obtained with the two search strategies, Dual and Master, as well as the default search strategy implemented in Solver. For each strategy, five values are displayed: $Z(INIT)$, the objective function value of the integer solution obtained after the initialization phase; $Z(LPCG)$, the objective function value of the best solution to the restricted LP relaxation of the master problem obtained at the end of the column generation phase (recall that we stop this phase after 40 iterations); $Z(IPBB)$, the value of the best integer solution obtained at the end of the branch-and-bound phase; Avg.CPU/iter, the average CPU time in seconds spent per iteration during the column generation phase; Avg.Fail/iter, the av-

Table 1. Comparing the Default, the Dual and the Master Search Strategies

Strategy	$Z(INIT)$	$Z(LPCG)$	$Z(IPBB)$	Avg.CPU/iter	Avg.Fail/iter
Default	28816	28816	28816	3.68	200.41
Dual	28816	12028	28816	1.84	0.12
Master	28816	13481	13952	2.28	37.27

erage number of failures per iteration when solving the employee subproblems during the column generation phase.

These results indicate that both the Dual and Master strategies improve significantly upon the default strategy. The Dual strategy quickly identifies negative reduced cost solutions, as shown by the low number of failures and the modest CPU time required. This strategy also exhibits the lowest objective value of $Z(LPCG)$, which indicates that it is the most effective for solving the LP relaxation. However, with the set of columns obtained after the column generation phase, the branch-and-bound algorithm is unable to identify an improving integer solution. By contrast, the Master strategy is less effective at solving the LP relaxation, but identifies a significantly better integer solution during the branch-and-bound phase.

Figure 1 presents, for the three search strategies, the evolution of the objective value during the column generation phase. The default strategy generates negative reduced cost solutions at every iteration but these added columns do not contribute to change the objective value. The Dual strategy is unable to improve the objective for the first 25 iterations, but then exhibits a sudden drop and a constant improvement at every iteration. These observations are consistent with the common knowledge in the column generation literature that the dual information is relatively poor during the first iterations. The Master strategy improves the objective value very early and then continues to make progress, although it is outperformed by the dual strategy in the last iterations. These results suggest that a hybrid method combining the Dual and Master strategies would be indicated to take advantage of the relative merits of both approaches.

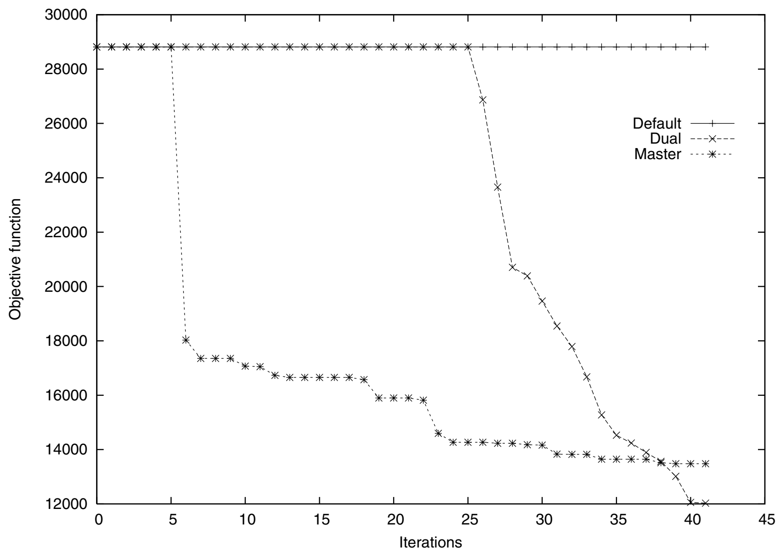

Fig. 1. Comparing the Strategies During Column Generation

6 Conclusion

In this paper, we have presented and compared two search strategies for solving the pricing subproblem in CP based column generation. The dual strategy aims to accelerate the solution of the LP relaxation of the master problem, while the master strategy drives the search towards integer solutions. We have presented computational results on an instance of a physician scheduling problem. These results show that these two strategies are promising and suggest that combining them might be even more effective. Further tests on other classes of problems would confirm the interest of such customized search strategies in CP based column generation. In addition, several other comparisons might be instructive:

– A comparison of the search strategies using the shortest path-based negative reduced cost constraint [6] rather than the simple pruning version used here;
– A comparison of the relative gains in terms of CPU time and branch-and-bound tree size obtained by sophisticated master problem branching rules (instead of the CPLEX default strategy used in our experiments) versus subproblem search strategies;
– An investigation of whether the subproblem search strategies can add to the gains achieved by more sophisticated master problem branching rules in a branch-and-price algorithm.

References

1. Barnhart, C., Johnson, E.L., Nemhauser G.L., Savelsbergh, M.W.P., Vance P.H.: Branch-and-Price: Column Generation for Solving Huge Integer Programs. Operations Research **46** (1998) 316–329
2. Beaulieu, H., Ferland, J.A., Gendron, B., Michelon, P.: A Mathematical Programming Approach to Scheduling Physicians in the Emergency Room. Health Care Management Science **3** (2000) 193–200
3. Bourdais, S., Galinier, P., Pesant, G.: HIBISCUS: A Constraint Programming Application to Staff Scheduling in Health Care. Principles and Practice of Constraint Programming, Proceedings of CP'03, Lecture Notes in Computer Science **2833** (2003) 153–167
4. Dantzig, G.B., Wolfe, P.: Decomposition Principle for Linear Programs. Operations Research **8** (1960) 101–111
5. Desrosiers, J., Dumas, Y., Solomon, M.M., Soumis, F.: Time Constrained Routing and Scheduling. Handbooks in Operations Research and Management Science **8**: Network Routing, edited by Ball, M.O., Magnanti, T.L., Monma, C.L., Nemhauser, G.L. (1995) 35–139 (North-Holland, Amsterdam)
6. Fahle, T., Junker, U., Karish, S.E., Kohl, N., Vaaben, N., Sellmann, M.: Constraint Programming Based Column Generation for Crew Assignment. Journal of Heuristics **8** (2002) 59–81
7. Fahle, T., Sellmann, M.: Cost Based Filtering for the Constrained Knapsack Problem. Annals of Operations Research **115** (2002) 73–93
8. Gilmore, P.C., Gomory, R.E.: A Linear Programming Approach to the Cutting Stock Problem. Operations Research **9** (1961) 849–859

9. Gilmore, P.C., Gomory, R.E.: A Linear Programming Approach to the Cutting Stock Problem: Part II. Operations Research **11** (1963) 863–888
10. Junker, U., Karish, S.E., Kohl, N., Vaaben, N., Fahle, T., Sellmann, M.: A Framework for Constraint Programming Based Column Generation. Principles and Practice of Constraint Programming, Proceedings of CP'99, Lecture Notes in Computer Science **1713** (1999) 261–274
11. Lübbecke, M.E., Desrosiers, J.: Selected Topics in Column Generation. Technical Report G-2002-64, GERAD, Montréal (2002)
12. Pesant, G.: A Filtering Algorithm for the Stretch Constraint. Principles and Practice of Constraint Programming, Proceedings of CP'01, Lecture Notes in Computer Science **2239** (2001) 183–195
13. Régin, J.C.: Generalized Arc Consistency for Global Cardinality Constraints. Proceedings of AAAI-96 (1996) 209–215 (AAAI Press/MIT Press)
14. Rousseau, L.-M., Gendreau, M., Feillet, D.: Interior Point Stabilization for Column Generation. Publication CRT-2003-39, Centre de recherche sur les transports, Université de Montréal (2003)
15. Rousseau, L.-M., Gendreau, M., Pesant, G., Focacci, F.: Solving VRPTWs with Constraint Programming Based Column Generation. Annals of Operations Research **130** (2004) 199–216
16. Savelsbergh, M.W.P.: A Branch-and-Price Algorithm for the Generalized Assignment Problem. Operations Research **45** (1997) 831-841
17. Sellmann, M., Zervoudakis, K., Stamatopoulos, P., Fahle, T.: Crew Assignment via Constraint Programming: Integrating Column Generation and Heuristic Tree Search. Annals of Operations Research **115** (2002) 207–225
18. Vanderbeck, F.: On Dantzig-Wolfe Decomposition in Integer Programming and Ways to Perform Branching in a Branch-and-Price Algorithm. Operations Research **48** 111–128
19. Vanderbeck, F., Wolsey, L.A.: An Exact Algorithm for IP Column Generation. Operations Research Letters **19** 151–159