Annals of Operations Research 139, 375–388, 2005

On Compact Formulations for Integer Programs Solved by Column Generation

DANIEL VILLENEUVE DVilleneuve@kronos.com
Kronos Inc., 3535 Queen Mary, suite 650, Montréal, H3V 1H8, Canada

JACQUES DESROSIERS * Jacques.Desrosiers@hec.ca
HEC Montréal and GERAD, 3000, chemin de la Côte-Ste-Catherine, Montréal, H3T 2A7 Canada

MARCO E. LÜBBECKE M.Luebbecke@math.tu-berlin.de
Technische Universität Berlin, Institut für Mathematik, Sekr. MA 6-1, Straße des 17. Juni 136, D-10623 Berlin, Germany

FRANÇOIS SOUMIS Francois.Soumis@gerad.ca
École Polytechnique de Montréal and GERAD, C.P. 6079, Succ. Centre-Ville, Montréal, H3C 3A7 Canada

Abstract. Column generation has become a powerful tool in solving large scale integer programs. It is well known that most of the often reported compatibility issues between pricing subproblem and branching rule disappear when branching decisions are based on imposing constraints on the subproblem's variables. This can be generalized to branching on variables of a so-called compact formulation. We constructively show that such a formulation always exists under mild assumptions. It has a block diagonal structure with identical subproblems, each of which contributes only one column in an integer solution. This construction has an interpretation as reversing a Dantzig-Wolfe decomposition. Our proposal opens the way for the development of branching rules adapted to the subproblem's structure and to the linking constraints.

Keywords: integer programming, column generation, branch-and-bound

Introduction

Branch-and-bound is a practically very successful generic method for solving mixed integer programs. It has been tailored to many particular applications, most notably by customized branching rules which exploit the respective problem structure. When the number of variables is huge, the bound at each node is obtained by column generation, that is, by iteratively adding variables via the questioning of an oracle (or subproblem or column generator, synonymously). The overall process, known as branch-and-price or integer programming column generation, hinges on what is called the compatibility of the branching rules with the oracle. Our discussion puts the notion of compatibility into a new perspective.

*Corresponding author.

Suppose a problem can be formulated in a *compact* way which explicitly reflects the oracle structure and a set of linking constraints. This type of formulation naturally leads to a solution by a decomposition process, such as the one proposed by Dantzig and Wolfe (1960). Therefore this problem can also be formulated in an *extensive* way, which results from the enumeration of a subset of solutions to the oracle. We make the point that the compact formulation does not only most naturally give rise to branching rules for the extensive formulation; its proper use also eliminates almost all difficulties with compatibility. Solving the compact formulation integrally is theoretically not different from solving any integer program, except for the way of computing the bounds by using the extensive formulation.

However, in particular in applications, it is not rare that only an extensive formulation and a pricing oracle are given. The contribution of this paper is to show, by construction, the existence of an associated compact formulation, and to demonstrate how it reduces compatibility issues.

Solving an extensive column generation formulation by way of a compact formulation can be seen as a straightforward, but useful complement of the use of the processes developed in the literature for decomposable integer programs, see e.g., Dantzig-Wolfe decomposition and Lagrangian relaxation. We outline how to algorithmically use the recovered compact formulation in integer programming column generation. The alternative we propose opens the way for the development of branching rules adapted to the oracle structure and to the linking constraints, in particular in a branch-and-cutcontext.

1. Column generation for integer programs

Consider the following program which we call the *compact formulation C*:

$$v(C) := \min\{\mathbf{c}^T\mathbf{x} \mid A\mathbf{x} = \mathbf{b},\ \mathbf{x} \in \mathcal{X}\}. \tag{1}$$

This is an integer program for $\mathcal{X} = \mathcal{P} \cap \mathbb{Z}^n_+$ where $\mathcal{P} \subseteq \mathbb{R}^n$ is a polyhedron. We remark that $\mathcal{X}$ could have a much more complicated non-linear definition. Without loss of generality, we assume that $v(C)$ be finite. Replacing $\mathcal{X}$ by $\operatorname{conv}(\mathcal{X})$ does not change $v(C)$. It is well known (see Schrijver, 1986) that we can represent each $\mathbf{x} \in \operatorname{conv}(\mathcal{X})$ as a convex combination of extreme points $\{\mathbf{p}_q\}_{q\in\mathcal{Q}}$ plus a non-negative combination of extreme rays $\{\mathbf{p}_r\}_{r\in\mathcal{R}}$ of $\operatorname{conv}(\mathcal{X})$, where the index sets $\mathcal{Q}$ and $\mathcal{R}$ are finite, i.e.,

$$\mathbf{x} = \sum_{q\in\mathcal{Q}} \mathbf{p}_q y_q + \sum_{r\in\mathcal{R}} \mathbf{p}_r y_r, \quad \sum_{q\in\mathcal{Q}} y_q = 1, \quad \mathbf{y} \in \mathbb{R}_+^{|\mathcal{Q}|+|\mathcal{R}|}.$$

Substituting for $\mathbf{x}$ in (1) and applying the linear transformations $c_j = \mathbf{c}^T\mathbf{p}_j$ and $\mathbf{a}_j = A\mathbf{p}_j$, $j \in \mathcal{Q} \cup \mathcal{R}$, we obtain an equivalent *extensive formulation* E_C of C:

$$\begin{aligned}
v(E_C) := \min \quad & \sum_{q\in\mathcal{Q}} c_q y_q + \sum_{r\in\mathcal{R}} c_r y_r \\
\text{subject to} \quad & \sum_{q\in\mathcal{Q}} \mathbf{a}_q y_q + \sum_{r\in\mathcal{R}} \mathbf{a}_r y_r = \mathbf{b} \\
& \sum_{q\in\mathcal{Q}} y_q = 1 \\
& \mathbf{y} \geq \mathbf{0} \\
& \mathbf{x} = \sum_{q\in\mathcal{Q}} \mathbf{p}_q y_q + \sum_{r\in\mathcal{R}} \mathbf{p}_r y_r \\
& \mathbf{x} \in \mathbb{Z}_+^n.
\end{aligned} \tag{2}$$

Typically, problem E_C has a large number $|\mathcal{Q}| + |\mathcal{R}| + n$ of variables, but possibly substantially fewer rows than problem C. Equation $\sum_{q\in\mathcal{Q}} y_q = 1$ is referred to as the *convexity constraint* over the extreme points of conv($\mathcal{X}$). This substitution easily generalizes to block diagonal matrices A, see Dantzig and Wolfe (1960). When we relax the integrality of $\mathbf{x}$, (2) becomes separable in $\mathbf{x}$ and $\mathbf{y}$, and we may also relax their linking constraints, obtaining a linear program in the variables $\mathbf{y}$ only. In general, requiring integrality of variables $\mathbf{y}$ does not lead to an integer program equivalent to C.

Alternatively, since the *original variables* $\mathbf{x}$ have to be integer (see also Holm and Tind, 1988), it is only natural to use them as the source of information in guiding branching and cutting decisions. Constraints representing these decisions on $\mathbf{x}$ are incorporated in C, either in the oracle structure or at the level of the linking constraints, and the decomposition process is then repeated. Problem E_C is then used only to compute a lower bound and to identify a solution in terms of the relaxation of the compact formulation.

Consider now the following integer program E formulated in an extensive way

$$\begin{aligned}
v(E) := \min \quad & \sum_{j\in J} c_j y_j \\
\text{subject to} \quad & \sum_{j\in J} a_{ij} y_j = b_i \qquad i \in I := \{1, \ldots, m\} \\
& y_j \in \mathbb{Z}_+ \qquad j \in J
\end{aligned} \tag{3}$$

the linear relaxation E' of which is supposedly solved by column generationusing a given pricing oracle in any form, regardless of whether mixed integer program, combinatorial algorithm, or other. Observe that in general E has no convexity constraint. We are not given an equivalent compact formulation on which we can analyze the solution of E', but we will show how to construct one. It has a block diagonal structure with identical sub-problems. We also propose a general separation strategy, based on imposing constraints on the oracle's domain.

We can always assume that the finite set J contains an index 0 for a dummy variable, i.e., $c_0 = 0$ and $\mathbf{a}_0 = \mathbf{0}$. It is later used as a slack variable in the reformulation C_E of E.

We denote by $A = \{(c_j, \mathbf{a}_j)\}_{j \in J}$ the set of all coefficient vectors. Note that this implies that A contains no duplicate elements, i.e., there is an obvious bijection between J and A. In column generation, the elements of A are accessed using an oracle. We think of it as a surjective function $\mathbf{f}$: $X \to A$ for some set X. The usefulness of the oracle relies on a smaller implicit description of X compared to A, and on knowing an algorithm to compute a negative reduced cost column as in

$$\min_{\mathbf{x} \in X} f_0(\mathbf{x}) - \sum_{i \in I} u_i f_i(\mathbf{x}),$$

where $\mathbf{u} \in \mathbb{R}^m$ and $\mathbf{f} = (f_0, (f_i)_{i \in I})$.

2. Solving *E* by branch-and-bound

2.1. Bounding

We obtain a lower bound on $v(E)$ from the linear programming relaxation E'. The reduced cost $\bar{c}_j(\mathbf{u}) = c_j - \sum_{i \in I} u_i a_{ij}$ of variable y_j is defined as a function on the dual variables $\mathbf{u} \in \mathbb{R}^m$. Customarily, according to Dantzig's *minimum reduced cost rule*, the oracle returns a vector $\mathbf{a} \in A$ corresponding to an index in $\arg\min_{j \in J} \bar{c}_j(\mathbf{u})$. Interestingly, not all elements $\mathbf{a} \in A$ can be generated this way. In the following example, if $c_2 > 1$, variable z_2 cannot be generated:

$$\begin{array}{rl} \min & z_1 + c_2 z_2 + \;\, z_3 \\ \text{subject to} & z_1 + \;\, 2z_2 + 3z_3 = 2 \\ & z_1, \quad\;\; z_2, \quad\; z_3 \in \mathbb{Z}_+. \end{array} \tag{4}$$

Given the dual variable $u \in \mathbb{R}$ associated with the equality constraint, z_2 is of minimum reduced cost if and only if $c_2 - 2u \leq 1 - u$ and $c_2 - 2u \leq 1 - 3u$, that is, if $c_2 \leq 1 - |u|$, in contradiction with $c_2 > 1$. It is a well known observation that the subset of generated columns may be integer infeasible. Here, we expose the stronger principal defect that even if feasibility could be ensured, sometimes we *cannot* obtain an optimal integer solution. This happens in (4) if $1 < c_2 < 2$: the unique optimal integer solution is $(z_1, z_2, z_3) = (0, 1, 0)$ of value c_2 while the solution restricted to the variables that can be generated is $(z_1, z_3) = (2, 0)$ of cost $2 > c_2$.

Similarly, Villeneuve (1999) illustrates that in Dantzig-Wolfe decomposition not all extreme points of the subproblem's polyhedron can be generated. We believe that analogous bad examples exist for other pricing rules as well. In conclusion, column generationmay not only be necessary after branching, but also branching may be necessary to generate the right columns. Clearly, column generationhas to be invoked not only at the root node but also at other nodes of the branch-and-boundtree. The next section shows various approaches reported in the literature which simultaneously take the pricing oracle's structure into account.

2.2. Branching

A branching rule partitions the solution space such that the current fractional solution is excluded, optimal integer solutions remain intact, and finiteness of the algorithm is ensured. The most immediate choice is to branch on variables of E. When the oracle supports excluding solutions which correspond to variables already down-branched on, this amounts to finding a kth best oracle solution instead of an optimal one (Sweeney and Murphy, 1979). This option is appealing because of its robustness against any restriction on J whatsoever. On the downside is the increased complexity of the oracle which grows proportionally to the total number of variables excluded at the current node, which can be far greater than the depth of this node in the tree. When the oracle is for instance a *shortest path* problem, a pricing algorithm must not return paths which belong to a set of forbidden paths (Villeneuve and Desaulniers, 2005). It may not always be as easy as that and without taking care, the oracle structure can be destroyed by branching decisions (see e.g., Johnson, 1989; Savelsbergh, 1997; Vance, 1998).

An alternative to branching on single fractional variables of E is to create two branches by imposing lower and upper bounds on a fractional *sum of variables*. When the oracle can be solved as an integer program, Vanderbeck (1994) proposes a strategy using a discretization of the oracle's domain. It is based on a rule originally developed by Ryan and Foster (1981) for set partitioning problems: Two rows must be covered either by one or by two distinct variables (see also Barnhart et al., 1998; Vanderbeck, 2000b). Vanderbeck and Wolsey (1996) generalize this to integer programs with general integer coefficients. Several other authors propose such a *branching on constraints*, either generic (branching on generalized upper bounds, or on special ordered sets, Nemhauser and Wolsey (1988)), or tailored to the respective oracle (e.g., Barnhart, Hane and Vance 1997; Chen and Powell 1999; Mehrotra, Murphy, and Trick, 2000; Mehrotra and Trick 1996; Ryan and Falkner, 1987; van den Akker, Hoogeveen, and van de Velde, 1999; Vanderbeck, 2000a). In general, the structure of the oracle has to be modified to take into account dual variables associated with the added branching constraints. Additional constraints may suffice, but more complex modifications may involve new constraints in E' as well as new binary variables and constraints in the oracle. The major drawback again is the increased size of the oracle integer program which might grow in some cases proportionally to the depth of the search tree.

The evident trade-off between a more elaborate branching rule and the resulting complication in the oracle leads to the notion of a *compatible* branching rule: The regeneration of variables which are forbidden in E (by whatever rule) has to be avoided while not increasing too much (better: at all) the oracle complexity with the growing depth of the search tree. When the oracle allows for constraints in the form of bounds we may as well impose branching decisions to the oracle's domain. This is a natural support for branching on the *original* variables of a compact formulation, as is often implicitly done in rules which branch on constraints, see the references above. When branching rules and the oracle are not considered separately but in an integrated way, based on a common structure, the notion of compatibility becomes void. In fact, branching decisions *reduce* the oracle's complexity until its solution becomes trivial. The relaxation E' is then used for the purpose of bounding only and for identifying a solution in terms of the variables of the compact formulation.

This approach was originally proposed by Desrosiers et al. (1984) for a *vehicle routing problem with time windows*. Interestingly, there are two types of branching strategies in that paper: One applies directly to the flow variables of the oracle while the other imposes cuts on the total cost and the number of vehicles utilized. The link between both types is indeed the compact formulation where one finds the original decision variables (network flow and time variables), some of them being part of the linking constraints and the oracle structure. Multicommodity flow formulations for various applications of vehicle routing and crew scheduling proposed by Desaulniers et al. (1998) are similar to this scheme. Several other authors also use the same idea, see e.g., Desrochers et al. (1991, 1995), Kohl et al. (1999), and Sol (1994).

3. A reformulation C_E of E

We construct a *compact formulation* C_E from which E follows by application of a decomposition process. More precisely, Dantzig-Wolfe decomposition applied to C_E yields an extensive formulation equivalent to E, however, with multiple column generators instead of the original single oracle we start with. Each generator contributes only one column in a final integer solution. We restrict ourselves to the case where X is closed and bounded, hence there exists a hyperbox B such that $X \subset B \subset \mathbb{R}^n$. We assume that we know a $\kappa \in \mathbb{Z}_+$ such that $\sum_{j \in J \setminus \{0\}} y_j \leq \kappa$ for a feasible solution $\mathbf{y}$, i.e.,

$$\sum_{j \in J} y_j = \kappa \quad \text{and} \quad y_0 = \kappa - \sum_{j \in J \setminus \{0\}} y_j. \tag{5}$$

We now formulate problem C_E which explicitly reflects the functioning of the oracle:

$$\begin{aligned} v(C_E) := \min \quad & \sum_{k \in K} f_0(\mathbf{x}^k) \\ \text{subject to} \quad & \sum_{k \in K} f_i(\mathbf{x}^k) = b_i \quad i \in I \\ & \qquad \mathbf{x}^k \in X^k \; k \in K := \{1, \ldots, \kappa\}. \end{aligned} \tag{6}$$

We define for each $X^k \subseteq X$ a subset $J^k \subseteq J$ of indices associated with the set $\mathbf{f}(X^k)$ of obtainable columns. We establish the connection between E and C_E via the following intermediate problem $\tilde{C}_E$, the variables y_j^k of which split the variables y_j of E in κ parts, i.e., $y_j = \sum_{k=1}^{K} y_j^k$:

$$\begin{aligned} v(\tilde{C}_E) := \min \quad & \sum_{k\in K}\sum_{j\in J^k} c_j y_j^k \\ \text{subject to} \quad & \sum_{k\in K}\sum_{j\in J^k} a_{ij} y_j^k = b_i \qquad i \in I \\ & \sum_{j\in J^k} y_j^k = 1 \qquad k \in K \\ & y_j^k \in \{0,1\}\ k \in K, \quad j \in J^k. \end{aligned} \tag{7}$$

Proposition 1. Problems C_E and $\tilde{C}_E$ are equivalent.

Proof. Since $\mathbf{f}$ is surjective, $|A| = |J|$ is finite, and $X^k \subseteq X$, $k \in K$, we may partition each X^k into a finite number of equivalence classes, defined by $\mathbf{x}_1^k \equiv \mathbf{x}_2^k$ if and only if $\mathbf{f}(\mathbf{x}_1^k) = \mathbf{f}(\mathbf{x}_2^k)$. Taking a single representative per equivalence class we obtain a finite subset $\tilde{X}^k \subseteq X^k$ which we use instead of X^k in (6). In order to reformulate C_E, let $\mathbf{x}_j^k \in \tilde{X}^k$ denote a representative of the equivalence class indexed by $j \in J^k \subseteq J$. By means of a binary linear combination of variables y_j^k we express each $\mathbf{x}^k \in \tilde{X}^k$, $k \in K$ by exactly one of its $|J^k|$ representatives:

$$\left\{ \mathbf{x}^k = \sum_{j\in J^k} y_j^k \mathbf{x}_j^k, \quad \sum_{j\in J^k} y_j^k = 1,\ y_j^k \in \{0,1\}, j \in J^k \right\} \quad k \in K. \tag{8}$$

Using (8) we can write this change of variables in (6) as follows:

$$\begin{aligned} \min \quad & \sum_{k\in K}\sum_{j\in J^k} y_j^k f_0\big(\mathbf{x}_j^k\big) \\ \text{subject to} \quad & \sum_{k\in K}\sum_{j\in J^k} y_j^k f_i\big(\mathbf{x}_j^k\big) &=& \ b_i \qquad i \in I \\ & \sum_{j\in J^k} y_j^k &=& \ 1 \qquad k \in K \\ & y_j^k &\in& \ \{0,1\} \quad k \in K, \quad j \in J^k \\ & \sum_{j\in J^k} y_j^k \mathbf{x}_j^k &=& \ \mathbf{x}^k \in \tilde{X}^k\ k \in K. \end{aligned} \tag{9}$$

At optimality constraints $\sum_{j\in J^k} y_j^k \mathbf{x}_j^k = \mathbf{x}^k \in \tilde{X}^k, k \in K$ are not necessary since exactly one $\mathbf{x}_j^k \in \tilde{X}^k$ is picked on the left-hand side by the binary variables. With $f_0(\mathbf{x}_j^k) := c_j$ and $f_i(\mathbf{x}_j^k) := a_{ij}$, formulation 9 obtained by a change of variables in problem C_E as given in 6 becomes identical to problem $\tilde{C}_E$ as given in (7). □

Formulation C_E is defined in such a way that it is valid for any branching decisions that will be made during branch-and-bound. This is why each $X^k \subseteq X$. The next proposition gives a sufficient condition to establish the equivalence between the root node problem C_E and E.

Proposition 2. If $X^k = X$ for all $k \in K$, problems $\tilde{C}_E$ and E are equivalent.

Proof. $X^k = X$ implies $J^k = J$. To complete the transformation, define variables $y_j = \sum_{k\in K} y_j^k$, $y_j \in \mathbb{Z}_+$. Summing up the κ convexity constraints of $\tilde{C}_E$ yields

$$\sum_{j\in J}\sum_{k\in K} y_j^k = \kappa \iff \sum_{j\in J} y_j = \kappa, \tag{10}$$

the last constraint being redundant by definition of κ in (5). Therefore $\tilde{C}_E$ becomes

$$\begin{aligned} \min \quad & \sum_{j\in J} c_j y_j \\ \text{subject to} \quad & \sum_{j\in J} a_{ij} y_j = b_i \qquad i \in I \\ & y_j \in \mathbb{Z}_+ \qquad j \in J \\ & y_j = \sum_{k\in K} y_j^k \qquad j \in J \\ & \sum_{j\in J} y_j^k = 1 \\ & y_j^k \in \{0, 1\} \qquad k \in K, \quad j \in J \\ & \sum_{j\in J^k} y_j^k \mathbf{x}_j^k = \mathbf{x}^k \in \tilde{X}^k \; k \in K. \end{aligned} \tag{11}$$

It remains to be shown that $\tilde{C}_E$ given in (11) is equivalent to E presented in (3). Clearly, any solution to (11) satisfies 3. On the other hand, we show that any solution to 3 in the non-negative integer variables y_j can be split into κ parts (allowing as many parts in variable y_0 as needed) which translate into variables y_j^k of (11). By (10), at most κ variables y_j are positive. Without loss of generality, let $y_0, y_1, \ldots, y_p$ be these variables, with $y_0 = \kappa - \sum_{j=1}^{p} y_j$. One possible assignment is:

$$y_0^k = \begin{cases} 1 & \text{if } 1 \le k \le y_0 \\ 0 & \text{otherwise,} \end{cases}$$

$$y_1^k = \begin{cases} 1 & \text{if } y_0 + 1 \le k \le y_0 + y_1 \\ 0 & \text{otherwise,} \end{cases}$$

$$\vdots$$

$$y_p^k = \begin{cases} 1 & \text{if} \sum_{j=0}^{p-1} y_j + 1 \le k \le \kappa \\ 0 & \text{otherwise,} \end{cases}$$

$$y_j^k = 0 \quad \text{if } j \notin \{1, 2, \ldots, p\} \quad \text{and} \quad 1 \le k \le \kappa.$$

Given this assignment, the $\mathbf{x}^k$-variables, $k \in K$ are trivially computed. Therefore problem E is a relaxation of $\tilde{C}_E$, and the assertion follows. □

More generally, an integral assignment can be done using the constraints of a transportation problem. Note that for a solution to the linear relaxation of E which is fractional in y_j, there exists a solution in y_j^k that satisfies the linear relaxation of $\tilde{C}_E$. Indeed, assume that y_j, $j \in J$, is a solution to E' and assign to y_j^k, $k \in K$, $j \in J$, the values:

$$y_j^k = \frac{y_j}{\kappa}, \quad k \in K, \quad j \in J.$$

By using (10), it is then easy to verify that this solution also satisfies $\tilde{C}'_E$.

4. Solving E using C_E

By Proposition 1, a lower bound on $v(C_E)$ is provided by $v(\tilde{C}'_E)$. Indeed, the linear relaxation $\tilde{C}'_E$ is based on the convexification of each subset X^k by using a convex combination of all feasible points in X^k instead of the binary linear combination used in Proposition 1.

At a node in the branch-and-boundtree, when the solution of $\tilde{C}'_E$ is integer, it replaces the incumbent solution if it improves on it. When $\tilde{C}'_E$ is fractional, we make use of $\mathbf{x}^k = \sum_{j \in J^k} y_j^k \mathbf{x}_j^k \in \tilde{X}^k$, $k \in K$ to reconstruct a solution in terms of the variables of problem C_E. Since $X \subset B$, we define $B^k := B$, $k \in K$ at the first node.

- If $\tilde{C}'_E$ is fractional and $\mathbf{x}^k \notin X^k$ for some $k \in K$, we create two new nodes by dividing the hyperbox B^k controlling the domain of X^k.
- If $\tilde{C}'_E$ is fractional and $\mathbf{x}^k \in X^k$ for all $k \in K$, then the *fractional* combination of solutions of X^k results in an integer solution for C_E. If $v(\tilde{C}'_E) = v(C_E)$, i.e.,

$$\sum_{k \in K} \sum_{j \in J^k} y_j^k f_0(\mathbf{x}_j^k) = \sum_{k \in K} f_0\left(\sum_{j \in J^k} y_j^k \mathbf{x}_j^k \right),$$

  the current solution is integer for C_E (and E), and we explore the remaining nodes.
- If $\tilde{C}'_E$ is fractional, $\mathbf{x}^k \in X^k$ for all $k \in K$, but $v(\tilde{C}'_E) < v(C_E)$, the difference is reflected by some index $k \in K$. This may occur if the cost function f_0 is non-linear. In this case, we also create two new nodes by dividing the hyperbox B^k.

By successive reductions of $B^k, k \in K$ this branching process allows for the exploration of all solutions of problem C_E, an equivalent compact version of the extensive formulation E. We note that branching strategies are not limited to the bisection of $B^k, k \in K$. This strategy serves as a simple illustration on how to produce a valid search tree where the oracle does not become more complicated as branching decisions accumulate.

Problem $\tilde{C}'_E$ may be huge, the number of variables compared to E being increased by a factor of κ, plus the κ convexity constraints. In the beginning, all subproblems are identical; hence a natural way to compute $v(\tilde{C}'_E)$ is to aggregate the variables and the convexity constraints, and to use a single oracle. Subproblems may differ only after branching, and the corresponding convexity constraints are introduced as needed.

5. Applications

In this section we present some applications, that is, the reconstruction of an adequate compact formulation on which branching and cutting strategies can easily be applied. Consider first the one-dimensional cutting stock problem, a classical example in column generation. Given are paper rolls of width W, and m demands b_i, $i \in I := \{1, \dots, m\}$ for orders of width w_i. The goal is to minimize the number of rolls to be cut into orders, such that the demand is satisfied. The standard extensive formulation by Gilmore and Gomory (1961), known for the strength of its linear relaxation bound, is

$$\min \left\{ \mathbf{1}^T \mathbf{y} \mid A\mathbf{y} \geq \mathbf{b},\ \mathbf{y} \in \mathbb{Z}_+^{|J|} \right\}, \tag{12}$$

where A encodes the set of $|J|$ feasible cutting patterns, i.e., $a_{ij} \in \mathbb{Z}_+$ denotes how often order i is obtained when cutting a roll according to $j \in J$. From the definition of the feasible patterns, condition $\sum_{i \in I} a_{ij} w_i \leq W$ must hold for every $j \in J$, and y_j determines how often the cutting pattern $j \in J$ is used.

Given the dual multipliers u_i, $i \in I$, the linear relaxation of (12) is classically solved via column generation, where the pricing oracle is:

$$\begin{array}{rl} \min & \left(x_0 - \sum_{i \in I} u_i x_i \right) \\ \text{subject to} & \sum_{i \in I} w_i x_i \leq W x_0 \\ & x_0 \in \{0, 1\} \\ & x_i \in \mathbb{Z}_+ \quad i \in I. \end{array} \tag{13}$$

In the above formulation, vector $\mathbf{x} = (x_0, (x_i)_{i \in I})$ separates in two parts: x_i, $i \in I$ is a non-negative integer variable that denotes the number of times order i is cut in a roll, and x_0 is a binary variable assuming value 1 if a roll is used and 0 otherwise. Note that when x_0 is set to 1, (13) is equivalent to solving a *knapsack problem* while if $x_0 = 0$, then $x_i = 0$ for all $i \in I$ and this null solution corresponds to an empty pattern, i.e., a roll that is not cut.

Our constructive procedure to recover a compact formulation equivalent to (12) leads to the definition of a specific subproblem for each roll. Let $K := \{1, \dots, \kappa\}$ be a set of rolls of width W such that $\mathbf{1}^T \mathbf{y} \leq \kappa$ for some feasible solution $\mathbf{y}$. Let $\mathbf{x}^k =$

$(x_0^k, (x_i^k)_{i \in I})$, $k \in K$, be duplicates of the **x**-vector, that is, x_0^k is a binary variable assuming value 1 if roll k is used and 0 otherwise, and x_i^k, $i \in I$ is a non-negative integer variable that denotes the number of times order i is cut from roll k.

Defining $f_0(\mathbf{x}^k) := x_0^k$ and $f_i(\mathbf{x}^k) := x_i^k$, $i \in I$ the compact formulation (6) reads as follows:

$$\begin{array}{llll} \min & \sum_{k \in K} x_0^k & & \\ \text{subject to} & \sum_{k \in K} x_i^k \geq b_i & i \in I & \\ & \sum_{i \in I} w_i x_i^k \leq W x_0^k & k \in K & \\ & x_0^k \in \{0,1\} & k \in K & \\ & x_i^k \in \mathbb{Z}_+ & k \in K, & i \in I. \end{array} \tag{14}$$

Formulation (14) of the cutting stock problem is due to Kantorovich (1960). It is known for the weakness of its linear relaxation. However, by applying an appropriate Dantzig-Wolfe decomposition, that is, in keeping the pricing oracle as an integer program given by (13), the lower bound provided by the linear relaxation of the resulting extensive formulation and that of 12 are the same. Branching decisions are then obviously taken on the **x**-variables.

It should be pointed out that existence of a compact formulation does not mean uniqueness. There exist alternative compact formulations for the cutting stock problem that give rise to the same linear relaxation of the extensive formulation. Valério de Carvalho (1999, 2002) proposes a clever network-based compact formulation in which the knapsack subproblem is solved as a particular minimum cost flow problem. Each subproblem path flow in that network gives a valid cutting pattern, and it corresponds to an extreme ray, except the null pattern which is the unique extreme point of the oracle's domain. We can turn the vast freedom of choosing a compact formulation into our advantage. For instance, we may wish to avoid symmetry (Barnhart et al., 1998) in the formulation. In fact, the process outlined in Section 4 is a good device in itself for reducing symmetry.

As a second type of application, consider column generation approaches used in the area of vehicle routing and crew scheduling. There the master problem is very often a *set partitioning* or a *set covering problem* while the oracle is given as a constrained shortest path problem. A path in the appropriate network represents a feasible itinerary for a vehicle, or for a crew. Therefore, it is only natural to define a specific subproblem for each vehicle or each crew member. This leads to (constrained) multicommodity flow problems as compact formulations, for which specialized branching and cutting decisions have been developed, see Desaulniers et al. (1998) for more details and the description of numerous branching and cutting rules. In the remainder we present two examples.

In Kohl et al. (1999), a subproblem is defined for each vehicle of the *vehicle routing problem with time windows*. The compact multicommodity flow formulation that is used allows cutting planes to be developed in terms of the network flow variables at the level of the linking constraints. These cuts generalize the subtour elimination constraints of the traveling salesman problem. Gamache et al. (1998) exploit in a special way the multicommodity formulation of their *airline rostering problem*. Here there is a subproblem for each pilot. The authors impose a very deep cut into selected subproblem domains. It does not only cut off the current infeasible fractional solution but at the same time it also removes from the oracle's domain a number of *integer* solutions.

Conclusion

Column generation and branch-and-boundare often reported to suffer from compatibility problems. Instead of considering these two components as separated we adopt a unifying perspective on solving extensive formulations by integer programming column generation. The principal advantage of solving an integer program by way of a compact formulation is to easily and directly exploit the structure of the oracle and that of the set of linking constraints. To a large extent the notion of compatibility becomes void in this framework. The small price we pay is the increased algorithmic administration of two concurrent formulations at a time.

Because of our rather general assumptions on the oracle there is quite some degree of freedom in obtaining a compact formulation, and imagination and experience are certainly helpful. Solving the compact formulation integrally is theoretically not different from solving any integer program, except for the way of computing the bounds by using the extensive formulation. This fundamental and indeed extremely simple approach has been in use now for almost twenty years (Desrosiers, Soumis, and Desrochers, 1984), and has been continually refined during this time.

Acknowledgments

We would like to thank Hatem Ben Amor, Guy Desaulniers, and Stefan Irnich for the time they spent in discussing a preliminary version of this work, and pointing us to inaccuracies in the presentation. We also appreciate the constructive comments by two anonymous referees.

References

Barnhart, C., C.A. Hane, and P.H. Vance. (1997). "Integer Multicommodity Flow Problems." *Lecture Notes in Economics and Mathematical Systems* 450, 17–31.

Barnhart, C., E.L. Johnson, G.L. Nemhauser, M.W.P. Savelsbergh, and P.H. Vance. (1998). "Branch-and-Price: Column Generation for Solving Huge Integer Programs." *Oper. Res.* 46(3), 316–329.

Chen, Z.-L. and W.B. Powell. (1999). "Solving Parallel Machine Scheduling Problems by Column Generation." *INFORMS J. Computing* 11, 78–94.

Dantzig, G.B. and P. Wolfe. (1960). "Decomposition Principle for Linear Programs." *Oper. Res.* 8, 101–111.

Desaulniers, G., J. Desrosiers, I. Ioachim, M.M. Solomon, F. Soumis, and D. Villeneuve. (1998). "A Unified Framework for Deterministic Time Constrained Vehicle Routing and Crew Scheduling Problems." In T.G. Crainic and G. Laporte (eds.), *Fleet Management and Logistics*, Norwell, MA, Kluwer, pp. 57–93.

Desrochers, M., J.K. Lenstra, M.W.P. Savelsbergh, and F. Soumis. (1991). "Vehicle Routing with Time Windows: Optimization and Approximation." In B.L. Golden and A.A. Assad, (eds.), *Vehicle Routing: Methods and Studies*, volume 16 of *Studies in Management Science and Systems*, North-Holland, pp. 65–84.

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

Desrosiers, J., F. Soumis, and M. Desrochers. (1984). "Routing with Time Windows by Column Generation." *Networks* 14, 545–565.

Gamache, M., F. Soumis, D. Villeneuve, J. Desrosiers, and E. Gélinas. (1998). "The Preferential Bidding System at Air Canada." *Transportation Sci.* 32(3), 246–255.

Gilmore, P.C. and R.E. Gomory. (1961). "A Linear Programming Approach to the Cutting-Stock Problem." *Oper. Res.* 9, 849–859.

Holm, S. and J. Tind. (1988). "A Unified Approach for Price Directive Decomposition Procedures in Integer Programming." *Discrete Appl. Math.* 20, 205–219.

Johnson, E.L. (1989). "Modelling and Strong Linear Programs for Mixed Integer Programming." In S.W. Wallace (ed.), *Algorithms and Model Formulations in Mathematical Programming*, Springer: Berlin, pp. 1–43.

Kantorovich, L.V. (1960). "Mathematical Methods of Organising and Planning Production." *Management Sci.* 6, 366–422. Translation from the Russian original, dated 1939.

Kohl, N., J. Desrosiers, O.B.G. Madsen, M.M. Solomon, and F. Soumis. (1999). "2-Path Cuts for the Vehicle Routing Problem with Time Windows." *Transportation Sci.* 33(1), 101–116.

Mehrotra, A., K.E. Murphy, and M.A. Trick. (2000). "Optimal Shift Scheduling: A Branch-and-Price approach. *Naval Res. Logist.* 47(3), 185–200.

Mehrotra, A. and M.A. Trick. (1996). "A Column Generation Approach for Graph Coloring." *INFORMS J. Comput.* 8(4), 344–354.

Nemhauser, G.L. and L.A. Wolsey. (1988). *Integer and Combinatorial Optimization*. Chichester: John Wiley & Sons.

Ryan, D.M. and J.C. Falkner. (1987). "A Bus Crew Scheduling System Using a Set Partitioning Mode." *Ann. Oper. Res.* 4, 39–56.

Ryan, D.M. and B.A. Foster. (1981). "An Integer Programming Approach to Scheduling." In A. Wren (ed.), *Computer Scheduling of Public Transport Urban Passenger Vehicle and Crew Scheduling*, Amsterdam: North-Holland, pp. 269–280.

Savelsbergh, M.W.P. (1997). "A Branch-and-Price Algorithm for the Generalized Assignment Problem." *Oper. Res.* 45(6), 831–841.

Schrijver, A. (1986). *Theory of Linear and Integer Programming*. Chichester: John Wiley & Sons.

Sol, M. (1994). *Column Generation Techniques for Pickup and Delivery Problems*. PhD thesis, Eindhoven University of Technology.

Sweeney, D.J. and R.A. Murphy. (1979). "A Method of Decomposition for Integer Programs." *Oper. Res.* 27, 1128–1141.

Valério de Carvalho, J.M. (1999). "Exact Solution of Bin-Packing Problems Using Column Generation and Branch-and-Bound." *Ann. Oper. Res.* 86, 629–659.

Valério de Carvalho, J.M. (2002). "LP Models for Bin-Packing and Cutting Stock Problems." *European J. Oper. Res.* 141(2), 253–273.

van den Akker, J.M., J.A. Hoogeveen, and S.L. van de Velde. (1999). "Parallel Machine Scheduling by Column Generation." *Oper. Res.* 47(6), 862–872.

Vance, P.H. (1998). “Branch-and-Price Algorithms for the One-Dimensional Cutting Stock Problem.” *Comput. Optim. Appl.* 9(3), 211–228.

Vanderbeck, F. (1994). *Decomposition and Column Generation for Integer Programs*. PhD thesis, Université catholique de Louvain.

Vanderbeck, F. (2000a). “Exact Algorithm for Minimising the Number of Setups in the One-Dimensional Cutting Stock Problem.” *Oper. Res.* 48(6), 915–926.

Vanderbeck, F. (2000b). “On Dantzig-Wolfe Decomposition in Integer Programming and Ways to Perform Branching in a Branch-and-Price Algorithm.” *Oper. Res.* 48(1), 111–128.

Vanderbeck, F. and L.A. Wolsey. (1996). “An Exact Algorithm for IP Column Generation. *Oper. Res. Lett.* 19, 151–159.

Villeneuve, D. (1999). “Logiciel de Génération de Colonnes.” PhD thesis, École Polytechnique de Montréal.

Villeneuve, D. and G. Desaulniers. (2005). “The Shortest Path Problem with Forbidden Paths.” *European J. Oper. Res.* 165(1), 97–107.