Available online at www.sciencedirect.com
SCIENCE DIRECT®
Operations Research Letters 34 (2006) 296–306
Operations Research Letters
www.elsevier.com/locate/orl

A generic view of Dantzig–Wolfe decomposition in mixed integer programming

François Vanderbeck[a,*], Martin W.P. Savelsbergh[b]

[a] *Université Bordeaux 1, 33405 Talence Cedex, France*
[b] *Georgia Institute of Technology, Atlanta, GA 30332-0205, USA*

Received 16 April 2005; accepted 26 May 2005
Available online 27 June 2005

Abstract

The Dantzig–Wolfe reformulation principle is presented based on the concept of generating sets. The use of generating sets allows for an easy extension to mixed integer programming. Moreover, it provides a unifying framework for viewing various column generation practices, such as relaxing or tightening the column generation subproblem and introducing stabilization techniques.

Keywords: Integer programming; Dantzig–Wolfe decomposition; Column generation; Stabilization techniques; Lagrangian relaxation

1. Introduction

Dantzig–Wolfe decomposition is a technique for dealing with linear and integer programming problems with embedded substructures that permit efficient solution. The technique has been applied successfully in a variety of contexts (for surveys see Barnhart et al. [1], Desaulniers et al. [4]). Implementing Dantzig–Wolfe decomposition based algorithms poses various challenges. The primary challenges revolve around the convergence of the dual bound computations and, in the context of integer programming, the enforcement of integrality restrictions.

The standard view of Dantzig–Wolfe decomposition is that it exploits the linear programming formulation of the Lagrangian dual. This so-called master linear program has an exponential number of variables that are handled using dynamic column generation. An alternative view is that Dantzig–Wolfe is a reformulation technique that gives rise to a mixed integer master program. The integrality restrictions of the original formulation translate into integrality restrictions in the reformulated problem. Viewing Dantzig–Wolfe as a reformulation technique allows for the development of a theoretical framework that facilitates handling of branching decisions and cutting planes in the master program.

This paper introduces a generic and unifying framework for Dantzig–Wolfe reformulation of mixed

* Corresponding author.
E-mail address: fv@math.u-bordeaux1.fr (F. Vanderbeck).

0167-6377/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.orl.2005.05.009

integer programs. The framework is based on the concept of generating sets. A major advantage of the framework is that it allows for the definition of Dantzig–Wolfe reformulations of mixed integer programs (it generalizes the reformulation technique for pure integer programs developed by Vanderbeck [18]). Furthermore, the flexibility of the framework can be exploited to implement a variety of column generation strategies for an efficient implementation of the method. We show that by appropriately modifying the set of generators, we can, for example, provide a warm-start, implement a primal-dual heuristic for the pricing problem, obtain stronger dual bounds, implement column re-optimization techniques, and incorporate stabilization techniques. As such, the framework captures and generalizes several ad hoc features proposed in the literature for specific optimization problems.

The paper is organized as follows. In Section 2, we review the Dantzig–Wolfe decomposition approach. We discuss when it can be applied, what value it brings, and what difficulties arise when implementing it. In Section 3, we introduce the concept of generating sets and present a unifying framework for the Dantzig–Wolfe reformulation of mixed integer programs. In Section 4, we discuss the potential benefits of aggregating generators, which corresponds to relaxing the pricing problem. In Section 5, we show how we may be able to improve dual bounds by restricting the set of generators, which corresponds to tightening the pricing problem. Finally, in Section 6, we introduce exchange vectors that can be incorporated into the set of generators to help stabilize column generation.

2. Dantzig–Wolfe decomposition

The Dantzig–Wolfe approach is an application of a decomposition principle: one chooses to solve a large number of smaller size, typically well-structured, subproblems instead of solving the original problem whose size and complexity are beyond what can be solved within a reasonable amount of time. The approach is well-suited for problems for which the constraint set admits a natural decomposition into subproblems defining more tractable combinatorial structures. Consider a mixed integer program of the form:

$$
\begin{aligned}
Z^{MIP} = \min \quad & c(x, y) && (1)\\
[MIP] \quad \text{s.t.} \quad & A(x, y) \geqslant a && (2)\\
& B(x, y) \geqslant b && (3)\\
& x \in \mathbb{R}^n_+ && (4)\\
& y \in \mathbb{N}^p, && (5)
\end{aligned}
$$

where $A \in \mathbb{Q}^{k\times(n+p)}$ and $B \in \mathbb{Q}^{l\times(n+p)}$ are rational matrices, and $c \in \mathbb{Q}^{(n+p)}$, $a \in \mathbb{Q}^k$ and $b \in \mathbb{Q}^l$ are rational vectors.

This structure encompasses cases in which the problem has:

1. Complicating constraints. The subsystem $B(x, y) \geqslant b$ represents a combinatorial optimization problem that can be solved much more efficiently than the global problem. The subsystem $A(x, y) \geqslant a$ represents a set of *complicating constraints*. For example, van den Akker et al. [16] apply Dantzig–Wolfe decomposition to a time-indexed formulation of a single machine scheduling problem in which B enforces that no two jobs can be processed simultaneously (B is an interval matrix) while A enforces that every job is processed exactly once. The subsystem B can be solved as the shortest path problem on an acyclic digraph.
2. Linking constraints. The subsystem $B(x, y) \geqslant b$ has a block diagonal structure while the subsystem $A(x, y) \geqslant a$ represents *linking constraints*. The classic example is the cutting stock problem where A enforces demand to be covered and B defines feasible ways to cut a wide roll into pieces of smaller width, i.e., there is a knapsack sub-problem for each wide roll: $\{y \in \mathbb{N}^n : \sum_i w_i y_i \leqslant W;\ 0 \leqslant y_i \leqslant d_i,\ \forall i\}$, where w_i denotes the width of item i and W that of the wide roll, while d_i is the demand for i (see for example Vanderbeck [17]).
3. Multiple subsystems. The subsystems $A(x, y) \geqslant a$ and $B(x, y) \geqslant b$ each represents a more tractable optimization problem (possibly having its own block diagonal structure) and the difficulty arises from having both of them simultaneously. (Some constraints can be present in both subsystems.) An example is the capacitated multi-item lot-sizing

problem, which takes the form

$$\min \quad \sum_{i,t} p_{it}\, x_{it} + \sum_{i,t} f_{it} y_{it} \tag{6}$$

$$\text{[CMILS]} \quad \text{s.t.} \quad \sum_{i} (x_{it} + s_i y_{it}) \leqslant C_t \quad \forall t \tag{7}$$

$$\sum_{\tau=1}^{t} x_{i\tau} \geqslant \sum_{\tau=1}^{t} d_{i\tau} \quad \forall i, t \tag{8}$$

$$x_{it} \leqslant c_{it} y_{it} \quad \forall i, t \tag{9}$$

$$x_{it} \geqslant 0, \quad y_{it} \in \{0, 1\} \quad \forall i, t, \tag{10}$$

where x_{it} is the quantity of item i that is produced in period t, y_{it} is 1 if the machine is set up for item i during period t, p_{it} and f_{it} are, respectively, production and setup costs, s_i is the capacity spent on a setup for item i, C_t is the capacity available in period t, d_{it} is the demand for product i in period t, and c_{it} is the maximum production level for i in period t. Then, (7) (9)–(10), can be viewed as a subsystem A that decomposes into a block for each period t defining a knapsack subproblem, while (8)–(10) can be viewed as a subsystem B that decomposes into a block for each item i defining a single-item lot-sizing subproblem.

There is a variety of ways in which such problem structure can be exploited to efficiently compute strong dual bounds around which an efficient exact optimization approach can be developed, e.g., apply Lagrangian relaxation or decomposition (dualizing linking constraints), employ cutting planes (efficient separation on the subproblem(s) is exploited to generate cuts), or use variable redefinition (a better formulation of the subproblem(s) is utilized to reformulate the original problem—see Martin [9]). The Dantzig–Wolfe decomposition approach can be viewed as a special form of variable redefinition (as presented by Vanderbeck [18] for the case of a pure integer program). The integer program can be reformulated in terms of variable weights associated with integer solutions and rays of the chosen subsystem polyhedron. The reformulation is traditionally called the *Master* program. Assuming a single and bounded subsystem B whose associated integer polyhedron is

$$X^B = \{y : By \geqslant b;\ y \in \mathbb{N}^p\}, \tag{11}$$

the *Master* takes the form

$$Z^M = \min \left\{ \sum_{q \in Q} c y^q \lambda_q : \sum_{q \in Q} A y^q \lambda_q \geqslant a; \right.$$
$$\left. \sum_{q \in Q} \lambda_q = 1;\ \lambda_q \in \{0, 1\}, \forall q \in Q \right\}, \tag{12}$$

where Q is the enumerated set of integer solutions to X^B, i.e., $X^B = \{y^q\}_{q \in Q}$, and λ_q is 1 if integer solution y^q is chosen and zero otherwise. The generalization to an unbounded subsystem or multiple subsystems is straightforward. The generalization to mixed integer programming is the subject of Section 3.

Due to its large number of variables, the linear programming (LP) relaxation of the master is solved using a column generation technique. For (12), the pricing problem takes the form

$$\zeta(\pi) = \min\{(c - \pi A) y : y \in X^B\},$$

where π are dual variables associated with constraints $\sum_{q \in Q} A y^q \lambda_q \geqslant a$. The column generation procedure starts by solving a master LP restricted to a subset of columns. While the pricing problem returns negative reduced cost columns, they are added to the linear program, which is then reoptimized. The intermediate master LP objective values are not valid dual bounds. However, a Lagrangian dual bound can readily be computed from the pricing problem objective value: applying Lagrangian relaxation to (12), dualizing the A constraints with weights $\pi \geqslant 0$, yields a valid bound of the form

$$L(\pi) = \pi a + \zeta(\pi). \tag{13}$$

Upon completion of the column generation procedure, this Lagrangian bound is equal to the master LP value. When this column generation algorithm is embedded in a branch-and-bound procedure to solve the integer program, then the overall algorithm is known as branch-and-price.

There are several reasons for considering Dantzig–Wolfe reformulations. First, it leads to a solution method that exploits our ability to efficiently solve the embedded subproblem. Second, it often leads to strong dual bounds. Lagrangian duality theory [6] tells us that solving the master LP is equivalent

to solving an LP on the intersection of the master constraints and the convex hull of the subproblem polyhedron. For (12) this means

$$Z^M_{\mathrm{LP}} = LD^A = \min\{cy : Ay \geqslant a;\ y \in conv(X^B)\}. \tag{14}$$

Thus, $Z^{MIP}_{\mathrm{LP}} \leqslant Z^M_{\mathrm{LP}} \leqslant Z^{MIP}$. The first inequality is typically strict unless $X^B_{\mathrm{LP}} = conv(X^B)$. Finally, when the decomposition is done around a subsystem B with a block diagonal structure and identical blocks, it avoids the symmetry that is naturally present in the original formulation. To be more precise, consider the block diagonal structure B,

$$B = \begin{pmatrix} B^1 & 0 & \dots & 0 \\ 0 & B^2 & \dots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \dots & B^K \end{pmatrix}, \tag{15}$$

with K identical blocks, i.e., $B^1 = B^2 = \cdots = B^K$. Then Dantzig–Wolfe reformulation takes the form

$$\min \left\{ \sum_{q\in Q} cy^q \lambda_q : \sum_{q\in Q} Ay^q \lambda_q \geqslant a;\ \sum_{q\in Q} \lambda_q = K;\ \lambda_q \in \mathbb{N}, \forall q \in Q \right\}.$$

This reformulation avoids symmetry, whereas, in the original formulation, variables must be indexed by $k = 1, \dots, K$ and a given solution can be represented in several ways by permuting the k indices.

Solving the Master LP poses several computational challenges (as discussed in [19]): slow convergence (the *tailing-off effect*), poor dual information at the outset (the *heading-in effect*), restricted master solution values that remain constant for several iterations (the *plateau effect*), dual solutions that jump from one extreme value to another (the *bang-bang effect*), and intermediate Lagrangian dual bounds that do not converge monotonically (the *yo-yo effect*).

Stabilization techniques have been developed to reduce these drawbacks (a review is given in [8]). Some, but not all, can be implemented in an LP framework. They can be classified into four categories (as presented in [19]): imposing bounds on dual prices, smoothing dual prices based on past information before passing them to the pricing problem, penalizing the deviation of the dual solution from a stability center (as in bundle methods [7]), and working with interior dual solutions rather than extreme point dual solutions. These techniques often have an intuitive interpretation in the primal space [2].

3. A unifying framework

In this section, we introduce a generic and unifying framework of Dantzig–Wolfe reformulation for mixed integer programs. The framework is based on the concept of generating sets. For each subsystem on which the decomposition is based, we define a finite set of generators, G, from which each subproblem solution can be generated. The variables λ_g of the reformulation represent the weights of the generators $g \in G$. To simplify the presentation, we assume a single subsystem

$$X^B = \{(x, y) : B(x, y) \geqslant b;\ x \in R^n_+;\ y \in \mathbb{N}^p\}. \tag{16}$$

However, the generalization to multiple subsystems is straightforward. We say that G^B is a *generating set for subsystem* X^B if the subsystem can be reformulated as

$$X^B = \left\{ (x, y) = \sum_{g\in G^B} g\lambda_g : \lambda \in R^B \right\},$$

where G^B must be a finite set even if X^B is not. Here, R^B represents specific restrictions on the weights λ_g, including integrality restrictions. Let R^B_{LP} denote the LP relaxation of R^B. The reformulation based on subsystem B takes the form

$$Z^M = \min \left\{ \sum_{g\in G^B} (cg)\lambda_g : \sum_{g\in G^B} (Ag)\lambda_g \geqslant a;\ \lambda \in R^B \right\}, \tag{17}$$

and the pricing problem encountered when solving the Master LP using column generation is

$$\zeta^B(\pi) \equiv \min\{(c - \pi A)g : g \in G^B\}. \tag{18}$$

In a standard Dantzig–Wolfe reformulation, the definitions of G^B and R^B ensure that:

Property 1 (*Lagrangian bound*). *The master linear programming relaxation offers a dual bound that is equal to the Lagrangian dual value, i.e.,*

$$\left\{(x, y) = \sum_{g \in G^B} g\lambda_g : \lambda \in R^B_{\mathrm{LP}}\right\} = conv(X^B).$$

Property 2 (*Integer reformulation*). *The master program* (17) *is a valid mixed integer reformulation of the original mixed integer program* (1)–(5).

In extensions of the Dantzig–Wolfe reformulation principle, which will be discussed in the following sections, one might consider defining generating sets for which these properties no longer hold.

This framework encompasses traditional definitions of Dantzig–Wolfe reformulation. In the *convexification* approach, the generating set is defined as the set of extreme points and rays of $conv(X^B)$. The resulting reformulation satisfies Property 1, but enforcing Property 2 demands to return to the original formulation space. For pure integer programs, an alternative approach that permits a true integer programming reformulation is to base the decomposition on the property that integer polyhedra can be generated from a finite set of feasible integer solutions plus a non-negative integer linear combination of integer extreme rays. This is the *discretization* approach presented by Vanderbeck [18]. It generalizes to mixed integer programs by applying discretization for the integer variables and convexification for the continuous variables.

In Table 1, we specify the definition of G^B and R^B for each of the above cases. We assume boundedness of the subsystem X^B and a single block in matrix B to simplify the notation, but the generalization to the unbounded case or a block diagonal subsystem is straightforward. Fig. 1 illustrates the reformulation of mixed integer programs, i.e., applying discretization for the integer variables and convexification for the continuous variables.

The distinction between convexification and discretization approaches is to be found in the expression of integrality restrictions: with the former, one must return to the original variables to express integrality. The discretization approach provides a framework in which it is more natural and more convenient to formulate branching constraints or cutting planes for the master [18,19]. Note that when the integer variables are binary, there is no difference between convexification and discretization.

In their study of column generation reformulations of the multi-item capacitated lot-sizing problem, Degraeve and Jans [3] observe that, for the reformulation based on convexification, enforcing integrality of the variables associated with the columns does not provide a valid integer programming reformulation. To remedy the situation, they consider a reformulation in which columns represent single-item production plans for specific feasible setup plans, which, in fact,

Table 1
Definitions of G^B and R^B under convexification and discretization approaches ((x^g, y^g) denotes the (x, y) solution defined by generator g)

Convexification:
$G^B = \{(x, y) \in \mathbb{R}^n_+ \times \mathbb{N}^p : (x, y) = \text{extreme point of } conv(X^B)\}$
$R^B = \{\lambda \geqslant 0 : \sum_{g \in G^B} \lambda_g = 1; \sum_{g \in G^B} y^g \lambda_g \in \mathbb{N}^p\}$

Discretization for an IP:
$G^B = \{y \in \mathbb{N}^p : By \geqslant b\}$
$R^B = \{\lambda : \sum_{g \in G^B} \lambda_g = 1; \lambda_g \in \mathbb{N},\ \forall g \in G^B\}$

Discretization for a MIP:
$G^B_p = \mathrm{proj}_y\, X^B = \{y \in \mathbb{N}^p : B(x, y) \geqslant b; x \geqslant 0\}$
$S^B(y) = \{x : x \text{ is an extreme point of } \{B(x, y) \geqslant b; x \in \mathbb{R}^n_+\}\}$
$G^B = \{(x, y) \in \mathbb{R}^n_+ \times \mathbb{N}^p : y \in G^B_p; x \in S^B(y)\}$
$G^B(y) = \{g = (x^g, y^g) : y^g = y \in G^B_p; x^g \in S^B(y)\}$
$R^B = \{\lambda : \sum_{g \in G^B} \lambda_g = 1; \sum_{g \in G^B(y)} \lambda_g \in \mathbb{N},\ \forall y \in G^B_p\}$

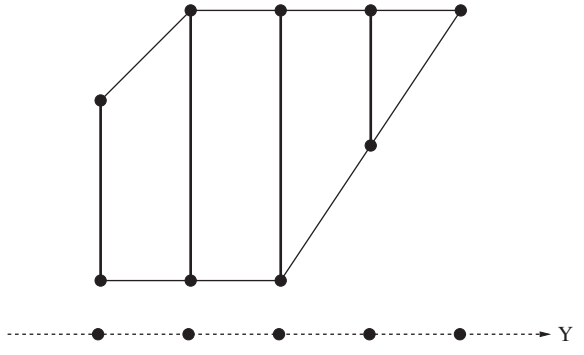

Fig. 1. Discretization for a MIP in dimension 2 ($n = p = 1$): the set of generators G^B (the bold lines represent $S^B(y)$ sets) and its projection in the y-space, G^B_p.

corresponds to an implementation of mixed integer programming reformulation based on discretization. Their study, performed independently, demonstrates that our proposed framework is not just of theoretical value.

Generating sets do not only provide a unifying framework for viewing Dantzig–Wolfe reformulation. They also point to a wider use of Dantzig–Wolfe reformulation. Considering generators that define a relaxed pricing problem can ease the solution of the pricing problem at the expense of getting weaker dual bounds. Section 4 presents relaxation strategies. On the other hand, considering generators that define a more restricted pricing problem can result in stronger dual bounds at the expense of spending more time to solve the pricing problem. Section 5 discusses restriction strategies. Expanding the set of generators with so-called exchange vectors enhances the stability of the Master LP solution process and is presented in Section 6.

4. Relaxing the definition of generators

A generating set defines a working space. We explore master solutions that can be defined in terms of the generators. The number of generators is typically huge. To ease the search for master solutions, we consider reducing the generating set by aggregating generators into what we will call *base-generators*. In a sense, a base-generator summarizes the main properties of the generators that are mapped onto it. Such a strategy is the analogue of the relaxation technique known in dynamic programming as *state space relaxation*, in which partial solutions represented by different states are aggregated into a single state. Relaxing the definition of generators to base-generators implies relaxing the pricing problem. This has computational advantages when generating base-generators can be done more efficiently. However, the dual bound obtained using base-generators may be weaker, i.e., Property 1 may be lost. Even worse, the resulting master may not be a valid reformulation of the original problem, i.e., Property 2 may also be lost.

For a general mixed integer program, one form of state space relaxation is to define the base-generators as the integer solutions of $\mathrm{proj}_y X^B$. Then, the set of base-generators, $\tilde{G}^B$, and the associated restriction system, $\tilde{R}^B$, take the form

$$\tilde{G}^B = G_p^B$$

$$\tilde{R}^B = \left\{ \lambda : \sum_{g \in G_p^B} \lambda_g = 1;\ \sum_{g \in G_p^B} B(x,g)\lambda_g \geqslant b;\ \lambda_g \in \mathbb{N},\ \forall g \in G_p^B \right\}. \tag{19}$$

When the definitions of G^B and R^B are replaced by $\tilde{G}^B$ and $\tilde{R}^B$ in (17), the resulting dual bound is weaker than that obtained with the original set of generators, as

$$\{A(x,y) \geqslant a;\ (x,y) \in conv(X^B)\} \subseteq \{A(x,y) \geqslant a;\ B(x,y) \geqslant b;\ y \in conv(\mathrm{proj}_y X^B)\}.$$

However, the master program defined in terms of base-generators offers a valid reformulation of the original problem, i.e., Property 2 still holds when the master (17) is defined with $\tilde{G}^B$ and $\tilde{R}^B$.

Let us see what happens when the above approach is applied to the CMILS problem. The set of base-generators corresponds to the set of feasible setup plans, $y \in \{0,1\}^T$, such that $\sum_{\tau=1}^{t} c_\tau y_\tau \geqslant \sum_{\tau=1}^{t} d_\tau$ for $t = 1, \ldots, T$. The latter constraints can be seen as a relaxation of constraints $\sum_{\tau=1}^{t} x_\tau \geqslant \sum_{\tau=1}^{t} d_\tau$. To obtain a feasible setup plan, it suffices to solve a discrete single-item lot-sizing problem with $x_t = c_t y_t$ instead of a continuous single-item lot-sizing problem with $x_t \leqslant c_t y_t$. Solving a discrete lot-sizing problem is easier than solving a continuous lot-sizing problem. A standard dynamic program for the discrete case has complexity $O(TD)$—where D is the total demand for the item considered—while it has complexity $O(TD^2)$ for the continuous case (if capacities are stationary, the respective complexities become $O(T^2)$ and $O(T^4)$). The dual bound may be weaker because the master LP allows additional solutions, as illustrated by the following example.

Example 1. Consider an instance with three time periods in which item i has demand $d_{i\cdot} = (0,1,1)$ and production capacities $c_{i\cdot} = (\frac{3}{2}, \frac{3}{2}, 1)$. Furthermore, consider generators $(x_i^a = (1,0,1), y_i^a = (1,0,1))$ and$(x_i^b = (0,1,1), y_i^b = (0,1,1))$. In the relaxed master LP (in which base-generators represent feasible setup plans y), the solution $x_{i\cdot} = (0,1,1)$ and $y_{i\cdot} = (\frac{1}{3}, \frac{2}{3}, 1)$

is feasible: take $\lambda_a = \frac{1}{3}$ for $y_i^a = (1, 0, 1)$ and $\lambda_b = \frac{2}{3}$ for $y_i^b = (0, 1, 1)$. However, this solution is not feasible in the original master LP (in which generators represent solutions to a continuous single-item lot-sizing problem). To get a fractional setup in the second period, the solution must involve columns in which $y_{i2} = 0$. But all columns in which $y_{i2} = 0$ have $x_{i1} \geqslant 1$. Hence, their presence in the convex combination implies a strictly positive production in the first period.

van Vyve [20] considers a related form of state space relaxation for the CMILS problem. After applying the variable redefinition approach discussed by Martin [9], van Vyve aggregates variables of the reformulation.

Next, we look at an example of state space relaxation in which it is not possible to maintain Property 2. For the CSP, the knapsack pricing problem can be relaxed by defining what Fekete and Schepers [5] call a dual feasible mapping.

Definition 1 (*Fekete and Schepers*). A mapping $u : w \in \mathbb{R} \to u(w) \in \mathbb{R}$ is dual feasible for a CSP instance if

$$\sum_i u(w_i) q_i \leqslant u(W) \quad \forall q \in Q,$$

where Q is the set of feasible knapsack solutions using the original weights w_i.

Fekete and Schepers give several examples of such mappings that yield non-trivial combinatorial dual bounds. Basically, they define weight buckets and map all items in that bucket onto a single weight. Given a dual feasible mapping u, base-generators are solutions to the knapsack problem with modified weights. Because the knapsack problem can be solved in pseudo-polynomial time, a mapping that reduces the magnitude of the weights makes the problem easier to solve. Moreover, the number of items is typically reduced, as several items see their width mapped onto the same value. Hence, several cutting patterns are associated with (mapped onto) a single base-pattern. Observe that an integer solution to the relaxed master program is not necessarily feasible for the original problem because the selected dual feasible cutting patterns may not satisfy the true knapsack constraint. Hence, Property 2 is lost. (We are not aware of any study in which dual feasible cutting patterns are applied in a column generation approach for the CSP.)

The straightforward use of the state space relaxation is to define columns as solutions to a relaxed pricing problem and to solve the resulting Master LP by a column generation procedure. However, the idea can be used in more subtle strategies. One can work with the relaxed pricing problem only at the outset of the column generation procedure to provide a *warm-start*. Observe that, for the projection framework (19), base-generators must be *lifted* to generator vectors (x, y) before switching to master formulation (17). In the CMILS example, this can be done by setting $x_t = c_t y_t$. In the CSP example, no lifting is needed. Generators obtained from solutions to the relaxed pricing problem are not necessarily feasible for the true pricing problem. Then, they are handled as "artificial variables" (that must be eliminated from the primal solution by progressively increasing their cost). They help to bound dual prices. Extending this idea further (with the above dual feasible mappings for the CSP in mind), a *scaling approach* can be implemented in which the pricing problem is refined iteratively.

Yet another way to exploit state space relaxation is to implement a 2-*stage* pricing problem solution approach in which the relaxed pricing problem is solved first, providing a dual bound. If it yields a negative reduced cost solution, then one attempts to generate a truly feasible solution based on that base-generator. For CMILS, generating a continuous setup production plan from a discrete setup production plan can be done in polynomial time. For a fixed y^*, a primal solution (x, y^*) can be computed by solving a network flow problem. On the other hand, a dual bound can be trivially obtained by choosing $x_t \in \{0, c_t y_t^*\}$ according to its reduced cost. Such two-stage approach can be viewed as a *primal-dual heuristic* for the pricing problem.

The concept of base-generators also suggests that one could price out dynamically all the columns for which this base-generator is a seed. Thus, before resorting to calling the pricing problem solver for the generation of new columns, one could *re-optimize* the columns currently in the pool by computing the best reduced cost column that can be obtained from a base-generator, assuming that this re-optimization is much cheaper than solving the pricing problem (as is the case for CMILS). This type of re-optimization strat-

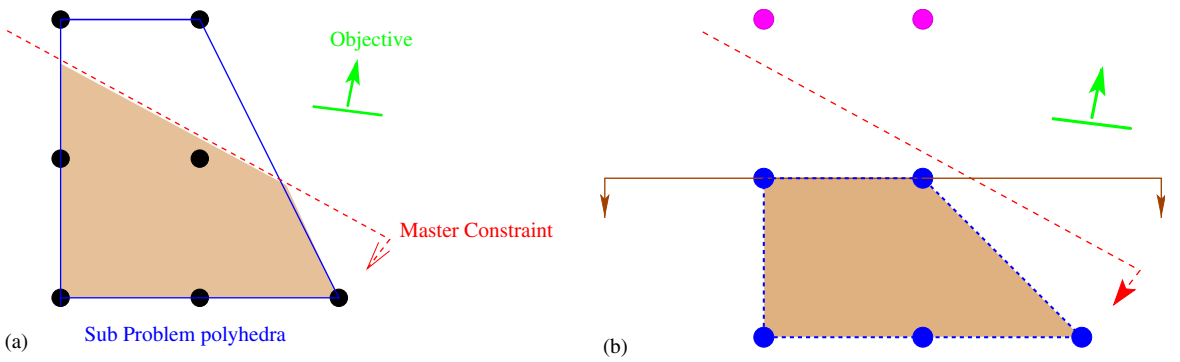

Fig. 2. The Lagrangian dual polyhedron (a) using all columns, (b) using proper columns.

egy can be used after just a few master simplex iterations, which allows the more expensive exact pricing problem solver to be used only when fully optimized dual prices are available. Re-optimization of existing columns has been used successfully before. For instance, Savelsbergh and Sol [13] apply local search to active columns.

5. Tightening the definition of generators

We now consider restricting the set of generators and hence the pricing problem. Generators that are not needed to express the feasible (or optimal) solutions to the master program can be removed from the generating set. In particular, under the discretization approach, a pricing problem solution that violates master program constraints can be discarded. Of course, insisting on generating pricing problem solutions that are feasible for the master program constraints is contrary to the decomposition principle, as it amounts to considering all the constraints in the pricing problem. However, one can make a step in this direction by considering, in the pricing problem, upper and lower bounds on the variables that are implied by the master constraints. Generators satisfying implied lower and upper bounds will be called *proper* generators. Enforcing pricing problem variable bounds can result in tighter dual bounds as illustrated in Fig. 2.

There are a variety of ways in which bounds on variables of the pricing problem can be obtained. First, problem knowledge may allow for refining of pricing problem variable bounds a priori. In the CSP, the number of pieces of a particular width in a cutting pattern should not exceed the demand for pieces of that width. In the CMILS, single-item production plans must satisfy the global capacity constraint in each period, i.e., $c_{it} \leqslant (C_t - s_i)$ in (9). In fact, even production of item i at level c_{it} might result in a situation where there is insufficient capacity to allow a feasible production plan for the remaining items. The production capacities c_{it} can be adjusted to account for the minimum capacity required by other items (see [10]). Second, bounds on the variables of the pricing problem can be derived by applying preprocessing techniques [12] to the original formulation. When subsystem B has a block diagonal structure with K identical blocks, the master constraints A imply bounds on aggregate variables (that represent the sum of K pricing problem variables), which in turn induce bounds on the disaggregated pricing problem variables (see [19]).

It can be effective to adjust pricing problem variable bounds dynamically. Consider, for example, a diving heuristic in a branch-and-price setting. Such a heuristic selects a specific branch at each node and re-optimizes the linear programs at every node using column generation. To avoid quickly ending up with an infeasible master program, it is vital to work with proper generators for the residual problem and thus to apply preprocessing at each node. Note that branching decisions may allow for more effective preprocessing and thus may induce tighter bounds.

Working with proper generators can produce tighter dual bounds, but it may also increase the complexity

of the pricing problem. Solving a pricing problem with bounded variables may be harder than solving its counterpart with unbounded variables. For the two-dimensional CSP, the unbounded knapsack pricing problem can be solved in pseudo-polynomial time by dynamic programming, whereas the bounded variant is strongly NP-hard. For the CMILS, the uncapacitated single-item lot-sizing problem is polynomially solvable, whereas the capacitated variant is NP-hard. A trade-off has to be made between the improved quality of the dual bound and the increased pricing problem complexity.

The generating set could be further restricted by eliminating *redundant* generators. A generator is *redundant* when the master mixed integer program admits an optimal solution that can be expressed without this generator. (Note in particular that all generators that are not proper are redundant.) Characterizing redundant generators, however, may not be easy. We can also define redundancy with respect to the master LP (as discussed, for example, by Lübbecke and Desrosiers [8]). Then, redundant generators correspond to constraints that are not facet defining for the dual master LP.

6. Exchange vectors as generators

Exchange vectors can be viewed as difference vectors between two feasible pricing problem solutions. Exchange vectors were introduced in the context of key formulations for solving Dantzig–Wolfe reformulations taking the form of a set partitioning problem with SOS constraints [1]. However, their use can be extended to a more general setting. Formally, exchange vectors can be defined as scaled rays of the homogeneous pricing problem. Assuming that B defines the pricing problem, let $C^B = \{r = (r_x, r_y) \in \mathbb{R}^n \times \mathbb{Z}^p : B(r_x, r_y) \geqslant 0\}$.

Definition 2. v is an exchange vector for (17) iff $\exists r \in C^B$, $\alpha \geqslant 0$, and $(x, y)^a$, $(x, y)^b \in X^B$ such that

$$v = \alpha r = (x, y)^a - (x, y)^b.$$

Consider introducing a set V of exchange vectors in the definition of the generating set. In the case of the discretization approach for an IP (see Table 1), this gives

$$G^B = \{y \in \mathbb{N}^p : By \geqslant b\} \cup V,$$

$$R^B = \left\{ \lambda : \sum_{g \in G^B \setminus V} \lambda_g = 1;\ \lambda_g \in \mathbb{N},\ \forall g \in G^B \right\}.$$

Exchange vectors, seen as "special columns", allow to define implicitly alternative pricing problem solutions that have not yet been generated: a single exchange vector can typically be applied to several columns of the current restricted master and hence define implicitly other columns that need not be generated explicitly. Viewed in the dual master program, exchange vectors define additional constraints linking the dual prices, which may help to stabilize the column generation procedure. (Valério de Carvalho [15] reports significant reductions in computing time using elementary exchange vectors for bin packing problems.)

In the CSP, where the pricing problem is a bounded integer knapsack problem, the set of feasible exchange vectors takes the form

$$V = \left\{ v \in \mathbb{Z}^n : \sum_i w_i v_i \leqslant 0; \right.$$

$$\left. \sum_i w_i \max\{v_i, 0\} \leqslant W;\ -d_i \leqslant v_i \leqslant d_i,\ \forall i \right\}, \quad (20)$$

i.e., a set of items i for which $v_i < 0$ is replaced by other items j for which $v_j > 0$ (the second constraint imposes the desired scaling). The associated cost is zero since the exchange does not introduce a new wide roll (see [11]). In the CMILS problem, the pricing problem is defined by constraints (8)–(10). Its solution by dynamic programming amounts to searching for a shortest path in an appropriately defined network. The difference between two pricing problem solutions can be seen as a circulation flow in this network (flow around cycles). Its cost represents the savings resulting from modifying the single-item production and setup plan. Exchange vectors for the CSP can also be seen as cycle flows in the network associated with the dynamic program for solving the knapsack pricing problem. In practice, exchange vectors that can be obtained as a linear combination of others can be ignored. For the above examples this means that only elementary cycles need to be considered.

Properties 1 and 2 may no longer hold when exchange vectors are introduced in the generating set. However, the resulting relaxation of the Master program can be controlled by appropriate restrictions on the set of exchange vectors used. The alternative is to treat the "undesirable" exchange vectors as artificial variables.

To prove that Property 1 still holds after the introduction of exchange vectors in the generating set, one must show that any LP solution to the augmented master can be obtained without using exchange vectors. Similarly, Property 2 still holds if any integer solution to the augmented master has an equivalent solution that makes no use of exchange vectors. For the CSP, Valério de Carvalho [15] shows that such equivalent solutions exist when the set of exchange vectors V is restricted to vectors with $\sum_i \min\{v_i, 0\} = -1$ (one item is being replaced by a set of smaller items). Perrot [11] shows that when the master reformulation is based on proper generators, the set of exchange vectors V must be further restricted to one-to-one exchanges, i.e., $\sum_i \max\{v_i, 0\} = -\sum_i \min\{v_i, 0\} = 1$, to maintain Property 1, but that it is sufficient to restrict exchange vectors to those with $\sum_i \min\{v_i, 0\} = -1$ to maintain Property 2.

Observe that column re-optimization, as discussed in Section 4, can be viewed as an implicit way to handle exchanges.

7. Final remarks

Varying the definition of the generating set amounts implicitly to examining different subsystems and dualized constraints. The quality of the dual bounds resulting from different decompositions could be compared theoretically a priori. Such a theoretical analysis is carried out for instance by Perrot [11] for the CSP and Thizy [14] for the CMILS problem. Thizy shows that only three different dual bounds can be obtained through Lagrangian duality. However, he considers only subsystems from the original set of constraints (while allowing for some constraints to be present in both subsystems A and B). What this paper points out is that the variety of Dantzig–Wolfe reformulations and associated dual bounds is typically wider. What matters most in practice is the trade-off between the duality gap and the computational effort required to solve the master and pricing problems.

Acknowledgements

We are grateful to the anonymous referees and to M.E. Lübbecke for providing comments and suggestions that have helped to improve the paper and streamline the presentation. F. Vanderbeck gratefully acknowledges support from INRIA through a Collaborative Research Action (ARC).

References

[1] C. Barnhart, E.L. Johnson, G.L. Nemhauser, M.W.P. Savelsbergh, P.H. Vance, Branch-and-price: column generation for huge integer programs, Oper. Res. 46 (1998) 316–329.
[2] O. Briant, C. Lemaréchal, Ph. Meurdesoif, S. Michel, N. Perrot, F. Vanderbeck, Comparison of bundle and classical column generation, INRIA Working paper 5453, January 2005.
[3] Z. Degraeve, R. Jans, A New Dantzig–Wolfe Reformulation and Branch-and-Price Algorithm for the Capacitated Lot Sizing Problem with Set Up Times, ERIM Report Series Research in Management, ERS-2003-010-LIS, 2003.
[4] G. Desaulniers, J. Desrosiers, M.M. Solomon (Eds.), Column Generation, Springer, Berlin, 2005.
[5] S.P. Fekete, J. Schepers, New classes of lower bounds for bin packing problems, Math. Programming 91 (2001) 11–31.
[6] A. Geoffrion, Lagrangian relaxation for integer programming, Math. Programming Study 2 (1974) 82–114.
[7] C. Lemaréchal, Lagrangian relaxation, in: M. Jünger, D. Naddef (Eds.), Computational Combinatorial Optimization, Springer, Berlin, 1998.
[8] M.E. Lübbecke, J. Desrosiers, Selected topics in column generation, Oper. Res. 2005.
[9] R.K. Martin, Generating alternative mixed-integer programming models using variable redefinition, Oper. Res. 35 (6) (1987) 820–831.
[10] K. Monneris, Problème de production par lots avec temps de mise en route par une méthode de branch-and-price, Rapport de stage de DEA, Université Bordeaux 1, juin 2002.
[11] N. Perrot, Integer programming column generation strategies for the cutting stock problem and its variants, Ph.D. Thesis, University Bordeaux 1 (2005).
[12] M.W.P. Savelsbergh, Preprocessing and probing techniques for mixed integer programming problems, ORSA J. Computing 6 (1994) 445–454.
[13] M.W.P. Savelsbergh, M. Sol, The general pickup and delivery problem, Transportation Sci. 29 (1) (1995) 17–29.
[14] J.-M. Thizy, Analysis of Lagrangian decomposition for the multi-item capacitated lot-sizing problem, INFOR 29 (4) (1991) 271–283.

[15] J.M. Valério de Carvalho, Using extra dual cuts to accelerate column generation, INFORMS J. Computing 17 (2) (2005).
[16] J.M. van den Akker, C.A.J. Hurkens, M.W.P. Savelsbergh, Time-indexed formulations for machine scheduling problems: column generation, INFORMS J. Computing 12 (1998) 111–124.
[17] F. Vanderbeck, Computational study of a column generation algorithm for bin packing and cutting stock problems, Math. Programming 86 (1999) 565–594.
[18] F. Vanderbeck, On Dantzig–Wolfe decomposition in integer programming and ways to perform branching in a branch-and-price algorithm, Oper. Res. 48 (1) (2000) 111–128.
[19] F. Vanderbeck, Implementing mixed integer column generation, in: G. Desaulniers, J. Desrosiers, M.M. Solomon (Eds.), Column Generation, Springer, Berlin, 2005.
[20] M. van Vyve, A solution approach of production planning problems based on compact formulations of lot-sizing models, Ph.D. Thesis, Université Catholique de Louvain, juin 2003.