Math Meth Oper Res (2002) 56:67–81

Mathematical Methods
of Operations Research

Primal cutting plane algorithms revisited

Adam N. Letchford[1], **Andrea Lodi**[2]

[1] Department of Management Science, Lancaster University, Lancaster LA1 4YW, England
(e-mail: A.N.Letchford@lancaster.ac.uk)
[2] D.E.I.S., University of Bologna, Viale Risorgimento 2, 40136 Bologna, Italy
(e-mail: alodi@deis.unibo.it)

Abstract. *Dual fractional* cutting plane algorithms, in which cutting planes are used to iteratively tighten a linear relaxation of an integer program, are well-known and form the basis of the highly successful *branch-and-cut* method. It is rather less well-known that various *primal* cutting plane algorithms were developed in the 1960s, for example by Young. In a primal algorithm, the main role of the cutting planes is to enable a feasible solution to the original problem to be improved. Research on these algorithms has been almost non-existent.

In this paper we argue for a re-examination of these primal methods. We describe a new primal algorithm for *pure 0-1 problems* based on strong valid inequalities and give some encouraging computational results. Possible extensions to the case of general mixed-integer programs are also discussed.

Key words: integer programming, cutting planes, primal algorithms

1 Introduction

Cutting plane algorithms, and more specifically branch-and-cut algorithms, have become very popular tools in recent years for solving integer and mixed-integer programs to optimality (see Nemhauser & Wolsey [20]; Padberg & Rinaldi [23]; Caprara & Fischetti [6]). To the knowledge of the authors, virtually all of the cutting plane algorithms in the modern literature are so-called *dual fractional* algorithms, based on the dual simplex method, in which the main role of the cutting planes is to iteratively tighten a linear relaxation of the original problem. These essentially have their roots in Gomory's classical *method of integer forms* (Gomory [12]).

What is less well-known is that there are two fundamentally different ways in which the simplex method and cutting planes can be used to solve integer

programs. The first alternative, which we call the *dual integral* approach, is to maintain integrality and dual feasibility throughout, and use cuts to drive the solution towards primal feasibility. The only algorithm of this type known to the authors is that of Gomory [14]. The second alternative, which we call the *primal* approach, is to maintain integrality and primal feasibility and use cuts to drive the solution towards dual feasibility. Algorithms of this third type, based on the *primal* simplex method, were devised by Ben-Israel and Charnes [5] and Young [28], and subsequently simplified by Glover [11] and Young [29].

An excellent introduction to, and comparison of, the three basic paradigms can be found in Garfinkel & Nemhauser [10]. However, whereas much effort has been devoted to improving the dual fractional framework, very little has been devoted to the dual integral and primal variants. (A prominent exception is a paper by Padberg & Hong [22], which we discuss in detail in Subsection 2.3.) The main goal of the present paper is to convince the reader that the third approach, the primal one, is viable.

The remainder of the paper is structured as follows. In Section 2 we introduce basic terms and briefly review the literature on primal cutting plane methods. In Section 3, we describe a modern primal cutting plane algorithm for *pure 0-1 problems* based on strong valid inequalities. We also consider possible extensions to general mixed-integer problems. In Section 4 we describe an implementation of this algorithm, and give some computational results which show that the new algorithm is a drastic improvement over Young's original method. Finally, conclusions are given in Section 5.

It should be noted that several other authors have recently become interested in primal algorithms for integer programming, see for example Schulz, Weismantel & Ziegler [24], Thomas [26], Urbaniak, Weismantel & Ziegler [27], Haus, Köppe & Weismantel [17], Firla et al. [9] and Eisenbrand, Rinaldi & Ventura [7]. But to our knowledge this is the first paper since Padberg & Hong [22] which attempts to devise a primal method based purely on cutting planes.

2 Primal cutting plane algorithms

2.1 Notation and definitions

Consider an *Integer Linear Program* (ILP) of the form:

$$\max\{c^T x : Ax \leq b, x \in Z_+^n\}.$$

Associated with the ILP are two polyhedra:

- $P := \{x \in \mathbb{R}_+^n : Ax \leq b\}$ (the feasible region of the linear programming relaxation),
- $P_I := \text{conv}\{x \in Z_+^n : Ax \leq b\}$ (the convex hull of feasible integer solutions).

A *cutting plane* is a linear inequality which is satisfied by all points in P_I, but which is not satisfied by all points in P.

Whereas a dual fractional cutting plane algorithm begins by solving the LP relaxation of the ILP, a primal cutting plane algorithm begins with a feasible

solution to the ILP, preferably a good one, which we will denote by $\bar{x}$. Moreover, it must be a solution with a very specific property: it must be an extreme point of both P and P_I. If no such $\bar{x}$ is known in advance, then artificial variables must be used to find one, much as in the ordinary primal simplex method. However, it is often easy to find one. In particular, for pure 0-1 problems any feasible solution gives a suitable starting vector $\bar{x}$.

The goal of the primal method is to iteratively improve the solution $\bar{x}$ by moving from one vertex of P_I to another, until no more improvement is possible and optimality is proven. (It can be viewed as an exact version of the well-known *local search* heuristic.) Of course, we do not have a description of P_I in advance, and this is why cutting planes are necessary.

This is how the method works. The simplex tableau associated with $\bar{x}$ is constructed. If $\bar{x}$ is dual feasible, it is optimal and the method stops. If not, then a non-basic variable with negative reduced cost is selected to come into the basis, and the usual primal simplex criterion is used to select a basic variable to leave the basis. Then a primal simplex pivot is made, leading to a new vector (or the same one in the case of degeneracy), which we will denote by x^*. If x^* is integral, then it represents an improved ILP solution, and it becomes the new $\bar{x}$. If on the other hand it is fractional, then a cutting plane is generated which cuts off x^*. Then, another attempt is made to pivot from $\bar{x}$, leading to a different x^*, and so on. The method terminates when $\bar{x}$ has been proved to be dual feasible.

2.2 Primal algorithms: Young's method

As mentioned above, in the 1960s several authors developed primal cutting plane algorithms for solving ILPs. For the sake of brevity we present only the version due to Young [29]. This version is simpler than the preceding ones, yet was shown by Young to be *finitely convergent* when an appropriate lexicographic pivot rule is used.

Young's method maintains integrality not only of $\bar{x}$, but of the *entire tableau* at all times. In order to do this, it only allows a pivot to be made if the pivot element is equal to 1. If the pivot element is not 1, then a cutting plane is added with the property that the pivot element in the enlarged LP tableau *is* 1.

The cutting planes of Young are defined as follows. Suppose that the pivot row of the simplex tableau takes the form:

$$x_i + \sum_{j \in NB} a_j x_j = x_i^*, \tag{1}$$

and x_k, for some $k \in NB$, wants to come into the basis in place of x_i. By assumption, all of the a_j are integer, as is x_i^*, and $a_k > 1$. Then the Young cut is

$$\sum_{j \in NB} \lfloor a_j / a_k \rfloor x_j \leq \lfloor x_i^* / a_k \rfloor. \tag{2}$$

After adding a new slack variable (say, s) to the Young cut and adding it to the tableau, we now find that it can serve as an alternative pivot row. Indeed, a_k can now come into the basis in place of s and the value of the pivot element is $\lfloor a_k / a_k \rfloor = 1$.

For the sake of clarity we now present a simple example of Young's algorithm in operation.

Example 1: Consider the following trivial ILP with two integer variables x_1, x_2 and two slack variables s_1, s_2:

$$\begin{aligned} \max \quad & f = 5x_1 + 2x_2 \\ \text{Subject to:} \quad & 2x_1 + x_2 + s_1 = 3 \\ & -2x_1 + x_2 + s_2 = 0 \\ & x \in Z_+^2,\ s \in Z_+^2. \end{aligned}$$

An obvious initial basic, primal feasible and integral solution is obtained by making s_1 and s_2 basic, to give $\bar{x} := (0, 0)$ and the corresponding initial simplex tableau is as follows.

		x_1	x_2	s_1	s_2
$-f$	0	5	2	0	0
s_1	3	2	1	1	0
s_2	0	−2	1	0	1

The standard primal simplex rule dictates that x_1 should be made basic in place of s_1. The first constraint therefore becomes the pivot row, leading to a pivot element with value 2. Since the pivot element is not 1, we generate the cutting plane

$$\lfloor 2/2 \rfloor x_1 + \lfloor 1/2 \rfloor x_2 + \lfloor 1/2 \rfloor s_1 \leq \lfloor 3/2 \rfloor$$

or, equivalently, $x_1 \leq 1$. Appending the slack variable s_3 this results in the following equation being added to the tableau:

$$x_1 + s_3 = 1.$$

Now we can make x_1 basic in place of s_3, to yield the following tableau:

		x_1	x_2	s_1	s_2	s_3
$-f$	−5	0	2	0	0	−5
s_1	1	0	1	1	0	−2
s_2	2	0	1	0	1	2
x_1	1	1	0	0	0	1

The new $\bar{x}$ is $(1, 0)$ and the profit has increased from 0 to 5. Next, we should make x_2 basic in place of s_1. This time the pivot element is 1 so we do not need to generate a Young cut. The next tableau is therefore:

		x_1	x_2	s_1	s_2	s_3
$-f$	-7	0	0	-2	0	-1
x_2	1	0	1	1	0	-2
s_2	1	0	0	-1	1	4
x_1	1	1	0	0	0	1

The new $\bar{x}$ is $(1, 1)$ and the profit has increased from 5 to 7. Moreover it is dual feasible, and hence optimal.

The Young cuts (2) can actually be regarded as a variant of the well-known *Gomory fractional cuts* (see Gomory [12] and Nemhauser & Wolsey [20]). Indeed, consider what *would* have happened if we *had* performed the pivot to bring x_k into the basis. The row (1) would have been divided by the pivot element to yield:

$$a_k^{-1} x_i + \sum_{j \in NB} (a_j/a_k) x_j = x_i^*/a_k. \tag{3}$$

Generating a Gomory fractional cut from (3) gives:

$$a_k^{-1} x_i + \sum_{j \in NB} f(a_j/a_k) x_j \geq f(x_i^*/a_k), \tag{4}$$

where $f(\cdot)$ denotes fractional part. Using the equation (3), this is easily shown to be equivalent to (2).

This means that, geometrically, the Young cuts have the effect of cutting off the fractional vector x^* which *would* have been obtained if a primal simplex pivot *had* been performed. That is, they remove points which are *adjacent* to $\bar{x}$ on the current polyhedron. Of course, they do not cut off $\bar{x}$ – this would not be sensible, given that $\bar{x}$ might be optimal. Instead they are frequently *tight* (satisfied at equality) at $\bar{x}$.

Unfortunately, the algorithm of Young typically leads to long sequences of degenerate pivots in which, although cuts are being added, $\bar{x}$ stays the same. Modifications have been suggested by, e.g., Arnold & Bellmore [1], [2], [3] and Sharma & Sharma [25], but the resulting algorithms are not competitive with dual fractional cutting plane algorithms (and certainly not with branch-and-cut).

2.3 Primal algorithms: the Padberg-Hong method

As mentioned above, research on primal cutting plane algorithms has been almost completely inactive. An important exception is a paper by Padberg & Hong [22], which describes a primal cutting plane algorithm for the *Symmetric Travelling Salesman Problem* (STSP). (A summary of this paper also appears in Padberg & Grötschel [21], sections 1.4 and 4.2.)

The approach of Padberg and Hong differs significantly from the older algorithms of Young et al. in several ways. First, they begin with a *subset* of the system $Ax \leq b$, rather than the entire system, which is of exponential size in the case of the STSP. Second, they use only *facet-defining* cuts, such as *subtour elimination*, *2-matching*, *comb* and *chain* inequalities. These cuts are generated dynamically by *constraint identification algorithms*. Third, they also allow the addition of several cuts at each iteration, instead of only one. All three

of these features are now standard in dual fractional algorithms, but none of them were permitted in Young's method. Moreover, whereas Young's method maintains integrality of the *entire* simplex tableau at all times, Padberg and Hong require only that the vector $\bar{x}$ be integral.

Finally, whereas Young's method is guaranteed to find the optimal solution (eventually), Padberg and Hong's method does not do so. It can reach the point where $\bar{x}$ is not dual integral, x^* is not a tour, yet no more violated facet-inducing cuts can be found. In this case, rather than branch or use general-purpose cuts, Padberg and Hong simply solve the current linear programming relaxation to optimality, using primal simplex pivots, in order to obtain a bound which can be used to assess the quality of the best tour found.

The constraint identification algorithms of Padberg and Hong are essentially equivalent to what are called *separation algorithms* in the modern literature (see Nemhauser & Wolsey, [20] and Grötschel, Lovász & Schrijver [15]). However, they are tailored to the primal context. Whereas a standard separation algorithm searches for a valid inequality which cuts off a fractional vector x^*, the algorithms used by Padberg and Hong add the extra requirement that the inequality found be *tight* (satisfied at equality) at $\bar{x}$. That is, they solve the following problem:

> **The Primal Separation Problem for a Class of Inequalities:** *Given some class $\mathcal{F}$ of inequalities which are valid for P_I, some $x^* \in P$ and some $\bar{x}$ which is an extreme point of both P and P_I, find a member of $\mathcal{F}$ which is violated by x^* and tight for $\bar{x}$, or prove that none exists.*

Presumably their intuition for doing this was as follows. If one were to add a cut which is violated by x^* but *not* tight at $\bar{x}$, then the next point encountered ($\hat{x}$, say) would simply be a convex combination of $\bar{x}$ and x^*. Given that the STSP is a 0-1 problem, $\hat{x}$ would still be fractional and no progress would have been made towards either finding an improving solution or proving optimality (dual feasibility) of $\bar{x}$. This argument is valid for general 0-1 ILPs, and also for mixed 0-1 problems, but it is less clear for problems with general integer variables, because when general integer variables are present it is possible for a convex combination of $\bar{x}$ and x^* to be integral even when x^* is not.

It was later shown by Padberg & Grötschel [21] that the extra restriction that the inequality be tight at $\bar{x}$ does not create any difficulties. Indeed, they showed that an inequality solves the primal separation problem if and only if the same inequality solves the standard separation problem for the point $\varepsilon x^* + (1 - \varepsilon)\bar{x}$, where ε is some small positive quantity whose encoding length is polynomial in the encoding length of the inequalities defining P_I. The same argument can be easily adapted to general mixed-integer programs.

Interestingly, the reverse does not hold: there is no obvious way to transform the standard separation problem into the primal version. This raises the possibility that, for some classes of inequalities, primal separation may be *easier* than standard separation. This is indeed the case; see Section 5.

3 A modern primal cutting plane algorithm

3.1 Basic framework

In limited experiments which we have performed with Young's algorithm, we have found that a huge number of cuts is generated, with coefficients growing

in an exponential manner, so that the LP solver encounters precision problems and crashes, sometimes before even a single non-degenerate pivot has been made. This happens even for relatively small instances (say, 10 variables and 5 constraints). This means that Young's algorithm, elegant though it is, is useless from a practical viewpoint.

In the light of the Padberg-Hong paper, and also the literature on dual fractional methods (see Nemhauser Wolsey [20], Padberg & Grötschel [21]), the key to producing a viable primal method would appear to be the use of *separation algorithms* to generate *strong* cutting planes dynamically. As said in Subsection 2.3, in the case of 0-1 ILPs there is a strong case for using *primal* separation algorithms, in order to generate cutting planes which are tight at $\bar{x}$. The situation is not so clear for mixed problems, or for problems with general integer variables, so for simplicity of exposition we begin with the pure 0-1 case. Moreover, we initially assume that the constraint matrix $Ax \leq b$ is small enough to be stored explicitly in the initial LP. More complicated problems are dealt with in the following subsections.

Our basic algorithmic framework is the following, which can be shown to converge finitely under certain mild assumptions:

- Step 1: Find a 'good' initial $\bar{x}$ and construct an appropriate tableau. (We do not require that the entire tableau be integral.)
- Step 2: If $\bar{x}$ is dual feasible, stop.
- Step 3: Perform a primal simplex pivot. If it is degenerate, return to 2.
- Step 4: Let x^* be the new vector obtained. If x^* is integral, set $\bar{x} := x^*$ and return to 2.
- Step 5: Call primal separation for known strong inequalities. If any are found, pivot back to $\bar{x}$, add one or more cuts to the LP, and return to 3.
- Step 6: Generate one or more general-purpose cuts (such as Gomory fractional or mixed-integer cuts), add them to the LP, pivot back to $\bar{x}$ and return to 3.

An interesting thing here is that, for pure 0-1 ILPs, we can easily generate a Gomory mixed-integer cut in step 6 which is guaranteed to solve the primal separation problem, i.e., to be tight at $\bar{x}$. For the sake of brevity we do not prove this in detail, but here is the basic idea. Consider a mixed-integer cut generated from the row of the simplex tableau associated with variable x_i, where x_i^* is fractional. It is well-known that this cut is a strengthened version of a so-called *intersection cut* derived from the disjunction $(x_i \leq 0) \vee (x_i \geq 1)$ (see for example Balas, Ceria and Cornuéjols [4]). The intersection cut is by definition tight at all n points which can be obtained by increasing the value of a non-basic variable until x_i becomes equal to either 0 or 1. One of these points is $\bar{x}$ itself. Therefore the intersection cut is tight at $\bar{x}$. The mixed-integer cut, being at least as strong, is therefore also tight at $\bar{x}$.

3.2 Enhancements

The next step is to extend this basic scheme to cope with problems (such as the STSP) in which the system $Ax \leq b$ has a huge (perhaps exponential) number of rows. Following Padberg and Hong, and also by analogy with the dual fractional approach, this can be tackled by using only a subset of these inequalities to construct the initial LP relaxation and generating the remaining

ones dynamically via a primal separation algorithm. However, there is a small complication: we might encounter the situation where x^* is binary, primal separation fails, yet x^* violates an inequality in the system $Ax \leq b$ which is *not tight* for $\bar{x}$. The following simple example shows how this might happen.

Example 2: Consider the following ILP with three 0-1 variables x_1, x_2, x_3:

$$\begin{aligned} \max \quad & f = x_1 + x_2 + x_3 \\ \text{Subject to:} \quad & x_1 + x_2 \leq 1 \\ & x_1 + x_3 \leq 1 \\ & x_2 + x_3 \leq 1 \\ & x \in \{0,1\}^3. \end{aligned}$$

Suppose that $\bar{x} = (1,0,0)$ and that therefore only the first two constraints are tight. Suppose moreover that only these two constraints are present in the initial LP relaxation. One simplex tableau corresponding to $\bar{x}$ is:

		x_1	x_2	x_3	s_1	s_2
$-f$	-1	0	0	1	-1	0
x_1	1	1	0	1	0	1
x_2	0	0	1	-1	1	-1

Now suppose we choose to make x_3 basic in place of x_1. This yields the following tableau:

		x_1	x_2	x_3	s_1	s_2
$-f$	-2	-1	0	0	-1	-1
x_3	1	1	0	1	0	1
x_2	1	1	1	0	1	0

However, this corresponds to $x^* = (0,1,1)$. This clearly violates the third constraint, but the third constraint is not tight at $\bar{x}$ and therefore would not be generated by a primal separation algorithm.

On such occasions we would appear to have no choice but to call a *standard* (i.e., non-primal) separation algorithm for the system $Ax \leq b$. Note however that x^* is known to be 0-1, a fact which could be exploited to simplify the standard separation algorithm. (For example, in the case of the STSP, it is much easier to identify violated subtour elimination inequalities when x^* is integral than when x^* is arbitrary. One merely has to compute connected components in a graph.)

Due to the above complication, in order to solve pure 0-1 ILPs with $Ax \leq b$ huge it is necessary to change step 4 of the algorithm given in the previous subsection to:

– Step 4′: Let x^* be the new vector obtained. If x^* is integral *and feasible*, set $\bar{x} := x^*$ and return to 2.

It is also necessary to insert the following step between steps 5 and 6:

– Step 5b: If x^* is integral but not feasible, find an inequality in the system $Ax \leq b$ which is violated by x^* yet *not tight* at $\bar{x}$, add it to the LP and perform a single (dual) simplex pivot to arrive at a new (fractional) point $\hat{x}$ which is a convex combination of $\bar{x}$ and x^*. Set $x^* := \hat{x}$.

We know from the argument at the end of the previous subsection that, after step 5b is performed, a mixed-integer cut generated in step 6 will cut off the new (fractional) x^* and be tight at $\bar{x}$. Therefore it would also cut off the previous (integral) x^*, which implies that the non-tight inequality generated in step 5b can be discarded. (Intuitively, the mixed-integer cut can be regarded as a 'rotated' version of the cut found in step 5b – rotated in such a way as to solve the primal separation problem.)

Example 2 (continued): In step 5b we would generate the cut $x_2 + x_3 \leq 1$. Adding a slack variable and expressing it in terms of the non-basic variables yields $-2x_1 - s_1 - s_2 + s_3 = -1$. We add this to the tableau and perform a dual simplex pivot (making x_1 basic in place of s_3) to yield the following tableau:

		x_1	x_2	x_3	s_1	s_2	s_3
$-f$	$-3/2$	0	0	0	$-1/2$	$-1/2$	$-1/2$
x_3	$1/2$	0	0	1	$-1/2$	$1/2$	$1/2$
x_2	$1/2$	0	1	0	$1/2$	$-1/2$	$1/2$
x_1	$1/2$	1	0	0	$1/2$	$1/2$	$-1/2$

This corresponds to the point $\hat{x} := (1/2, 1/2, 1/2)$, which therefore becomes our new x^*. Then, regardless of whether we generate a fractional cut or mixed-integer cut in step 6, and regardless of the row chosen to generate it, we will obtain the cut $(1/2)s_1 + (1/2)s_2 + (1/2)s_3 \geq 1/2$. In terms of the original variables, this is $x_1 + x_2 + x_3 \leq 1$. This is tight at $\bar{x}$ and cuts off both the original invalid integer point $(0, 1, 1)$ and the new fractional point $(1/2, 1/2, 1/2)$. Therefore it solves the primal separation problem and the inequality $x_2 + x_3 \leq 1$ generated in step 5b can be discarded.

Now we consider other ways in which this basic framework can be enhanced (see Padberg & Rinaldi [23] and Caprara & Fischetti [6], for a description of these techniques in the dual fractional context). First, to avoid the basis becoming too large, we can use a *cut pool*. Whenever the slack of a cut becomes positive, we can remove the cut from the LP and place it in the pool. Each time an augmentation (i.e., an improvement) occurs in step 4, cuts which have become slack can be moved from the LP to the pool and cuts which have become tight can be moved from the pool to the LP. *Variable pricing* (i.e., starting with a subset of variables and introducing others dynamically when needed) can also be done easily.

However, *fixing variables* on the basis of reduced costs seems more prob-

lematic. This is because the primal approach as stated yields a sequence of *lower* bounds but not *upper* bounds. Of course, if for some reason the code has to be terminated before optimality has been proven, we can simply do what Padberg and Hong did, and solve the current LP to optimality to obtain an upper bound. This LP can be solved in a relatively small number of primal simplex pivots, because $\bar{x}$ can be used as a starting basis.

Finally, we briefly consider the possibility of *branching*. In order to obtain a method which is competitive with branch-and-cut (Padberg & Rinaldi [23]), it would be desirable to somehow integrate primal cutting plane methods with branch-and-bound. However, branching is problematic in the primal context. The point is that one wishes to cut off x^* but does not want to cut off the current best known feasible solution $\bar{x}$ (it may be optimal). If one tries to branch in the standard way, by choosing a fractional variable x_i^* and creating two subproblems (one with the constraint $x_i = 0$ added and the other with the constraint $x_i = 1$ added), $\bar{x}$ will be excluded from one of the two subproblems. This would mean that we had no starting basis on one of the branches.

What is really needed is a non-standard branching rule which removes x^* but leaves $\bar{x}$ intact on both sides. Here is an example of such a non-standard branching rule. Suppose that there is a pair of 0-1 variables x_i, x_j such that $\bar{x}_i = \bar{x}_j = 0$ and $0 < x_i^* < x_j^* \leq 1$. Create two branches, one with the constraint $x_i = 0$ added; the other with the constraint $x_i \geq x_j$ added. With this rule, no feasible solution is lost, $\bar{x}$ is feasible for both branches, and x^* is removed in both branches.

We know of some more general and powerful branching rules, which will be described in a future paper. In the present paper, however, we are mainly concerned with the cutting side.

3.3 Extension to mixed-integer problems

The approach outlined so far applies only to pure 0-1 ILPs and it is natural to ask if it can be generalized. In fact, it can be applied with little modification to *mixed 0-1* problems – i.e., problems in which variables are either continuous or binary. (We omit details for the sake of brevity.) However, it is much harder to deal with problems in which *general integer variables* are present. This is because, when such variables are present, it is possible for a convex combination of $\bar{x}$ and x^* to represent a feasible solution even when x^* does not.

Example 3: Consider the following ILP with two variables x_1, x_2:

$$\begin{aligned} \max \quad & f = x_1 + x_2 \\ \text{Subject to:} \quad & x_1 - x_2 \leq 0 \\ & x_1 + 3x_2 \leq 6 \\ & x \in Z_+^2. \end{aligned}$$

The fractional point $x^* = (3/2, 3/2)$ is adjacent to the integer point $\bar{x} = (0, 0)$, but the point $(2/3)x^* + (1/3)\bar{x} = (1, 1)$ is integral (and optimal).

This means that we can no longer guarantee that there is a valid inequality which is simultaneously tight for $\bar{x}$ and violated by x^*. That is, there may be no inequality which solves the primal separation problem.

Surprisingly, it turns out that primal separation algorithms suffice even in this case, at least in theory. Let us denote by $\hat{x}$ the point on the line connecting $\bar{x}$ and x^* which is a feasible MILP solution. If there are several candidates for $\hat{x}$, then choose the one which is closest to x^*. Then a simple geometric argument shows that there is a valid inequality which is violated by x^*, yet tight at $\hat{x}$. Such an inequality can be found (in theory) by calling a primal separation algorithm with $\hat{x}$ replacing $\bar{x}$ in the input. If such an inequality is obtained and added to the LP, then a new attempt to pivot from $\bar{x}$ will take us to $\hat{x}$. In this way we can augment successfully.

A more serious problem is *branching*. Recall that we desire a branching rule which will exclude a fractional point x^* but which will leave $\bar{x}$ intact on both branches. Whereas this is fairly easy to accomplish in the (mixed) 0-1 case, there is no obvious way of doing it when general integer variables are present. This is a challenging problem for future research.

4 Computational experiments

In this section we give the results of some computational experiments which show the improvement obtained by using the algorithm presented in Subsection 3.1 instead of Young's algorithm. Because Young's method breaks down for anything but very small instances, we have constructed a number of small, random 0-1 ILPs.

We have chosen to use *multi-dimensional 0-1 knapsack problems*, i.e., problems of the form $\max\{c^T x : Ax \leq b, x \in \{0,1\}^n\}$, where $c \in Z_+^n$, $A \in Z_+^{m \times n}$ and $b \in Z_+^m$. These were chosen for two reasons. First, we have an obvious initial $\bar{x}$, since the origin is always feasible. Second, there is a simple well-known class of strong cutting planes, the so-called *lifted cover inequalities* (LCIs) – see for example Nemhauser & Wolsey [20]. Suppose that an individual constraint has the form $\sum_{i=1}^n a_i x_i \leq b$ and suppose that $C \subseteq \{1, \ldots n\}$ is a *cover*, i.e., that $\sum_{i \in C} a_i > b$. Then the cover inequality $\sum_{i \in C} x_i \leq |C| - 1$ is valid and can be *lifted* to become facet-inducing for the 0-1 knapsack polytope $conv\{x \in \{0,1\}^n : \sum_{i=1}^n a_i x_i \leq b\}$, though not necessarily for the multi-dimensional knapsack problem itself.

It was proved in Ferreira, Martin & Weismantel [8] that (standard) separation of cover inequalities is $\mathcal{NP}$-hard and, in our paper Letchford & Lodi [18] we prove that the primal version is too. Therefore we resort to a simple heuristic for primal separation based on the ideas in Gu, Nemhauser & Savelsbergh [16]. We just pick each individual knapsack constraint in turn and insert items into C in non-increasing order of x^* value until C is a cover. However, we ensure that exactly one member $j \in C$ satisfies $\bar{x}_j = 0$, to ensure that the cover inequality generated is tight at $\bar{x}$. Then we lift the inequality in a greedy way to obtain an LCI which is tight at $\bar{x}$. All violated LCIs found are added to the LP.

This is how we generated our test problems. For $n \in \{5, 10, 15, 20, 25\}$ and $m \in \{5, 10\}$, we constructed five random instances, making 50 instances in total. The objective function coefficients are integers generated uniformly between 1 and 10. For the instances with $m = 5$, the left-hand side coefficients

Table 1. Results with Young's method

	n	Opt	Proven	Aug.	Overall Cuts	on-line %GAP	off-line %GAP
D	5	5	5	2.0	6.0	0.00	0.00
	10	2	1	4.6	127.0	7.32	3.50
	15	1	0	6.8	102.4	10.56	6.56
	20	0	0	8.6	184.6	9.16	5.93
	25	0	0	11.8	85.2	6.92	4.08
S	5	5	4	0.8	1436.0	6.26	0.00
	10	4	3	4.2	1151.2	11.62	3.08
	15	3	0	6.6	390.2	15.38	3.62
	20	0	0	8.6	136.0	14.52	6.17
	25	1	0	11.6	99.2	12.04	4.29

are also integers generated uniformly between 1 and 10. For the instances with $m = 10$, the left-hand side coefficients have a 50% chance of being an integer generated uniformly between 1 and 10, but also have a 50% chance of being zero. That is, these instances are sparse. In all cases the right-hand side of each constraint was set to half the sum of the left hand side coefficients. (This is well-known to lead to non-trivial instances of the multi-dimensional 0-1 knapsack problem.)

We found that more complex instances (e.g., with larger coefficients, more constraints or variables, or higher density) caused Young's algorithm to run into numerical difficulties and crash.

We used the CPLEX 7.0 callable library for performing primal simplex pivots. For all algorithms, we handled the upper bounds of 1 *implicitly* (see, for example, page 39 of Nemhauser & Wolsey [20]). This led to a significant reduction in both memory and running time, but the cost was a rather complicated checking of cases when selecting the variables to enter and leave the basis, and also a need to check sign conventions carefully when generating cuts.

The results with Young's original algorithm are displayed in Table 1. The letters 'D' and 'S' indicate dense and sparse instances respectively. The number of variables is given in the column marked 'n'. The column 'Opt' shows the number of times (out of five) the optimal solution was found without numerical problems due to the integers in the tableau becoming huge. (We set a limit of one million for this purpose.) The column 'Proven' shows the number of times the optimal solution was found and *proven* to be optimal. The following two columns show the average number of successful augmentations and the average number of Young cuts added before either proving optimality or exiting due to huge coefficients. The column headed 'on-line %GAP' gives the average percentage gap between the best lower bound and the upper bound obtained by solving the final LP to optimality. The column headed 'off-line %GAP' gives the average percentage gap between the best lower bound and the actual integer optimum (found by the CPLEX Mixed-Integer solver).

Table 2 shows the analogous figures for our primal cutting plane algorithm, but there is an extra column showing the average number of LCIs generated.

Already it is apparent that our proposed algorithm is superior to Young's original scheme. However, we found that the algorithm can be improved fur-

Table 2. Improved version with LCIs

	n	Opt	Proven	Aug.	Overall Cuts	primal LCIs	on-line %GAP	off-line %GAP
	5	5	5	2.0	5.2	4.4	0.00	0.00
	10	5	5	4.8	13.8	8.6	0.00	0.00
D	15	3	2	7.2	390.6	12.0	5.68	4.12
	20	5	2	10.0	380.2	28.6	1.38	0.00
	25	3	2	13.2	477.4	21.2	1.98	0.76
	5	5	5	0.8	5.0	5.0	0.00	0.00
	10	5	5	4.6	10.0	9.0	0.00	0.00
S	15	5	4	6.8	113.0	11.6	1.06	0.00
	20	3	2	9.4	376.8	24.6	6.68	3.14
	25	1	1	11.8	400.6	30.6	6.32	2.18

Table 3. Results with augmentation heuristic added

	n	Opt	Proven	Aug.	Overall Cuts	primal LCIs	on-line %GAP	off-line %GAP
	5	5	5	2.0	5.8	4.4	0.00	0.00
	10	5	5	4.8	13.8	8.6	0.00	0.00
D	15	5	4	8.0	112.2	16.6	0.68	0.00
	20	5	2	10.0	422.2	34.2	1.28	0.00
	25	4	2	13.2	466.2	26.6	1.94	0.57
	5	5	5	0.8	5.0	5.0	0.00	0.00
	10	5	5	4.6	10.2	9.0	0.00	0.00
S	15	5	4	6.8	93.6	12.4	1.48	0.00
	20	5	2	10.2	343.2	30.6	2.20	0.00
	25	4	1	12.6	375.2	40.2	4.00	0.22

ther by the addition of a simple *heuristic* for augmentation, which is called whenever we have generated 25 Gomory cuts without success (i.e., without either augmenting or proving dual feasibility). The heuristic is to iteratively pick, among all indices j such that x_j^* is fractional, the index which minimizes $|x_j^* - \bar{x}_j|$, and to (temporarily) add to the LP the extra constraint $x_j = \bar{x}_j$. The effect of this is that, next time we pivot from $\bar{x}$, we will obtain a different x^* for which x_j^* is integral. Repeating this procedure, we often find that we arrive at an integral x^*, in which case we can augment successfully (and delete all of the extra constraints). This gave the results shown in Table 3.

In our view these results are very promising. It seems likely that the addition of branching to this scheme will lead to a viable algorithm for 0-1 programming. This will be the subject of a future paper.

5 Discussion and conclusions

We have examined the potential of primal cutting plane algorithms and argued that, just as dual fractional cutting plane algorithms can be improved by

using 'strong' cuts, cut pools, branching, etc., so can primal algorithms. We have also provided preliminary computational results to back this claim.

Two other recent papers provide further motivation for working on primal algorithms. First, in our companion paper, Letchford & Lodi [18], we show that primal separation is often much easier than standard separation. Amongst other things, we give primal separation algorithms for the following inequalities which are faster than their corresponding standard equivalents by a *polynomial factor*:

- odd cycle inequalities for the stable set problem,
- subtour elimination constraints for the STSP,
- weak odd CAT inequalities for the Asymmetric TSP,
- simple comb inequalities for the STSP (a polynomial-time algorithm for the standard separation of these inequalities has been recently given by Letchford and Lodi [19]).

A second paper containing relevant results is that of Eisenbrand, Rinaldi & Ventura [7]. They show that a class of 0-1 ILPs can be solved in polynomial time if and only if the associated primal separation problem can. Moreover, they show that primal separation of *blossom* inequalities for capacitated *b*-matching problems is much easier than standard separation, although not necessarily asymptotically faster.

Acknowledgement. The authors would like to thank both Robert Weismantel and the anonymous referee for helpful comments.

References

1. Arnold LR, Bellmore M (1974) Iteration skipping in primal integer programming. Operations Research 22:129–136
2. Arnold LR, Bellmore M (1974) A generated cut for primal integer programming. Operations Research 22:137–143
3. Arnold LR, Bellmore M (1974) A bounding minimization problem for primal integer programming. Operations Research 22:383–392
4. Balas E, Ceria S, Cornuéjols G (1993) A lift-and-project cutting plane algorithm for mixed 0-1 programs. Mathematical Programming 58:295–324
5. Ben-Israel A, Charnes A (1962) On some problems of diophantine programming. Cahiers du Centre d'Études de Recherche Opérationelle 4:215–280
6. Caprara A, Fischetti M (1997) Branch-and-cut algorithms. In Dell'Amico M, Maffioli F, Martello S (eds.) Annotated Bibliographies in Combinatorial Optimization. Wiley, New York, pp. 45–64
7. Eisenbrand F, Rinaldi G, Ventura P (2002) 0/1 primal separation and 0/1 optimization are equivalent. 13^{th} Annual ACM-SIAM Symposium on Discrete Algorithms. San Francisco, January 6–8, 2002
8. Ferreira CE, Martin A, Weismantel R (1996) Solving multiple knapsack problems by cutting planes. SIAM Journal of Optimization 6:858–877
9. Firla RT, Haus U-U, Köppe M, Spille B, Weismantel R (2001) Integer pivoting revisited. Working paper, Institute of Mathematical Optimization, University of Magdeburg
10. Garfinkel RS, Nemhauser GL (1972) Integer programming. Wiley, New York
11. Glover F (1968) A new foundation for a simplified primal integer programming algorithm. Operations Research 16:727–740
12. Gomory RE (1958) Outline of an algorithm for integer solutions to linear programs. Bulletin of the AMS 64:275–278

13. Gomory RE (1960) An algorithm for the mixed-integer problem. Report RM-2597, Rand Corporation (Never published)
14. Gomory RE (1963) An all-integer programming algorithm. In Muth JF, Thompson GI (eds.) Industrial Scheduling. Prentice-Hall, New York
15. Grötschel M, Lovász L, Schrijver AJ (1988) Geometric algorithms and combinatorial optimization. Springer, Berlin
16. Gu Z, Nemhauser GL, Savelsbergh MWP (1998) Lifted cover inequalities for 0-1 integer programs: computation. INFORMS Journal on Computing 10:427–437
17. Haus U-U, Köppe M, Weismantel R (2001) The integral basis method for integer programming. Mathematical Methods of Operations Research 53:353–361
18. Letchford AN, Lodi A (2001) Primal separation algorithms. Technical Report OR-01-5, D.E.I.S., University of Bologna
19. Letchford AN, Lodi A (2001) Polynomial-time separation of simple comb inequalities. Technical Report OR-01-8, D.E.I.S., University of Bologna
20. Nemhauser GL, Wolsey LA (1988) Integer and combinatorial optimization. Wiley, New York
21. Padberg MW, Grötschel M (1985) Polyhedral computations. In Lawler E, Lenstra JK, Rinnooy Kan A, Shmoys D (eds.) The Traveling Salesman Problem. John Wiley & Sons, Chichester, pp. 307–360
22. Padberg MW, Hong S (1980) On the symmetric travelling salesman problem: a computational study. Mathematical Programming Study 12:78–107
23. Padberg MW, Rinaldi G (1991) A branch-and-cut algorithm for the resolution of large-scale symmetric travelling salesman problems. SIAM Review 33:60–100
24. Schulz A, Weismantel R, Ziegler G (1995) 0-1 integer programming: optimization and augmentation are equivalent. In Lecture Notes in Computer Science, vol. 979. Springer, Berlin, pp. 473–483
25. Sharma S, Sharma B (1997) New technique for solving primal all-integer linear programming. Opsearch 34:62–68
26. Thomas R (1995) A geometric Buchberger algorithm for integer programming. Mathematics of Operations Research 20:864–884
27. Urbaniak R, Weismantel R, Ziegler G (1997) A variant of Buchberger's algorithm for integer programming. SIAM Journal on Discrete Mathematic 1:96–108
28. Young RD (1965) A primal (all-integer) integer programming algorithm. Journal of Research of the National Bureau of Standards 69B:213–250
29. Young RD (1968) A simplified primal (all-integer) integer programming algorithm. Operations Research 16:750–782