Finite Disjunctive Programming Characterizations for General Mixed-Integer Linear Programs

Binyuan Chen

Department of Systems and Industrial Engineering, University of Arizona, Tucson, AZ 85720, bychen@email.arizona.edu

Simge Küçükyavuz, Suvrajeet Sen

Department of Integrated Systems Engineering, The Ohio State University, Columbus, OH 43210,
kucukyavuz.2@osu.edu, sen.22@osu.edu

In this paper, we give a finite disjunctive programming procedure to obtain the convex hull of general mixed-integer linear programs (MILP) with bounded integer variables. We propose a finitely convergent *convex hull tree algorithm* which constructs a linear program that has the same optimal solution as the associated MILP. In addition, we combine the standard notion of sequential cutting planes with ideas underlying the convex hull tree algorithm to help guide the choice of disjunctions to use within a cutting plane method. This algorithm, which we refer to as the *cutting plane tree algorithm*, is shown to converge to an integral optimal solution in finitely many iterations. Finally, we illustrate the proposed algorithm on three well-known examples in the literature that require an infinite number of elementary or split disjunctions in a rudimentary cutting plane algorithm.

Key words: Mixed-integer programming, disjunctive programming, convex hull, finite convergence.

1. Introduction

Mixed-integer linear programming (MILP) has come a long way. Advanced software (e.g. XPRESS, CPLEX etc.) are routinely solving MILP problems with thousands of variables with very reasonable computational demands. The area is now growing towards mixed-integer nonlinear programming, as well as stochastic MILP. In both cases, impressive computational results have been reported by Bonami et al. (2008) and Yuan and Sen (2009), respectively.

One of the more important lessons learned from MILP and related literature is that valid inequalities are indispensable in robust MILP software. Thus, it is common for commercial software to include both general purpose valid inequalities, such as Gomory cuts and mixed-integer rounding cuts (see Cornuéjols (2008) for a recent survey), and special purpose valid inequalities, such as flow

cover inequalities (Padberg et al. 1985) and flow path inequalities (Van Roy and Wolsey 1985), within branch-and-cut methods. As a result, the study of valid inequalities remains an important part of the MILP literature, even though the performance of *pure* cutting plane methods leave a lot to be desired.

Over the past several decades, the polyhedral study of binary MILP has progressed along several avenues. Disjunctive and lift-and-project cuts (Balas 1979, Balas et al. 1993), semidefinite relaxations (Lovász and Schrijver 1991) and reformulation-linearization technique (RLT) (Sherali and Adams 1990, 1994) have provided alternative approaches to generate cutting planes that define the convex hull of feasible points of a binary MILP. The case for MILP with general integers has not been as well understood, even when the integer variables are bounded. While pure integer programming can be shown to have finite representation using Gomory cuts (Gomory 1963), the same is not true for general MILP. Recently, Adams and Sherali (2005) provided a generalization of the RLT methodology to the case of general MILP problems using Lagrange Interpolation Polynomials to compute the bound factors in the RLT process. Thus, formation of the convex hull of integer points in MILP problems using RLT has been resolved. However, the same cannot be said about disjunctive programming methods.

While disjunctive cuts are natural to use for general MILP problems, a cutting plane procedure using disjunctive cuts has not been proved to be finitely convergent. Indeed, the facial disjunctive property (Balas 1979) was deemed critical for finite convergence and this property holds for binary MILP, but not for general MILP. Unfortunately, as shown in Figure 2 of Sen and Sherali (1985), the absence of the facial disjunctive property could lead to infinitely many iterations. Subsequently, Owen and Mehrotra (2001) provided the proof that in the absence of the facial disjunctive property, a one-variable-at-a-time method for convexifying disjunctive sets leads to an infinite convergent process that ultimately does provide the convex hull of feasible points. Of course, one could write a binary expansion of each general integer variable and the resulting formulation is a mixed 0-1 program which can be sequentially convexified. However, Owen and Mehrotra (2002) illustrate that this approach is not practical.

In this paper, assuming a nonempty feasible set and bounded general integer variables (or a bounded optimal objective value), we address the following questions:

- Is it possible to use the disjunctive programming methodology to describe the convex hull of MILP solutions in finitely many steps without introducing binary variables?

- Is there a constructive methodology to obtain an optimal solution to a general MILP using a disjunctive programming characterization of its convex hull?

- If we are restricted to introduce only one cutting plane in any iteration, is there a finitely convergent disjunctive programming algorithm that solves a general MILP?

We provide answers to the above questions in the affirmative. Moreover, we provide preliminary evidence that our approach can be implemented in a manner that the convexification process is guided by the objective function, so that fewer inequalities are generated. However, we caution that it is best to incorporate these ideas within a more standard branch-and-cut methodology, and work along these lines will be reported in subsequent papers.

2. Convex Hull of Bounded General Mixed-Integer Programs

Consider a mixed-integer linear program with n variables, of which the first n_1 $(n_1 \leq n)$ are integer, and the remaining are continuous. Such a problem may be stated as

$$\min_{x \in X}\{c^T x \mid X = \{Ax \leq b, x \in \mathbb{Z}_+^{n_1} \times \mathbb{R}_+^{n-n_1}\}\}. \tag{1}$$

The set obtained by relaxing the integrality requirements of X is denoted by X_L. We assume that all integer variables are bounded with $x_j \in [0, u_j]$ for all $j = 1, \ldots, n_1$. Such problems will be referred to as general MILP with bounded integer variables.

Suppose now that each interval $[0, u_j]$ is divided into t_j sub-intervals $[\ell_{1j} := 0, u_{1j}]$, $[\ell_{2j}, u_{2j}], \ldots, [\ell_{t_j j}, u_{t_j j} := u_j]$, where $\ell_{\kappa_j j} \in \mathbb{Z}_+$ and $u_{\kappa_j j} \in \mathbb{Z}_+$, with $\ell_{\kappa_j j} \leq u_{\kappa_j j}$, define the left and right-hand sides of interval $\kappa_j \in \{1, \ldots, t_j\}$ for $j = 1, \ldots, n_1$, and $\ell_{\kappa_j+1 j} - u_{\kappa_j j} \leq 1$ for $\kappa_j \in \{1, \ldots, t_j - 1\}$ so that the sub-intervals span all integers in $[0, u_j]$. Given a partition $\mathcal{P}$, the collection of all n_1-tuples $\kappa := (\kappa_1, \ldots, \kappa_{n_1})$, where $\kappa_j \in \{1, \ldots, t_j\}$ for $j = 1, \ldots, n_1$, is denoted by $K(\mathcal{P})$. Then a

unit partition, $\mathcal{P}^*$, of all integer points is a partition for which $u_{\kappa_j j} - \ell_{\kappa_j j} \leq 1$, for all $\kappa_j = 1, \ldots, t_j$, and all $j = 1, \ldots, n_1$. For a given vector $\kappa \in K(\mathcal{P}^*)$, an index $j \in \{1, \ldots, n_1\}$, and a polyhedron $\bar{X}$, we define two sets $P^-(\kappa, j, \bar{X})$ and $P^+(\kappa, j, \bar{X})$ as follows:

$$P^-(\kappa, j, \bar{X}) := \{x \in \bar{X} | \ell_{\kappa_i i} \leq x_i \leq u_{\kappa_i i}, i = 1, \ldots, n_1; x_j \leq \ell_{\kappa_j j}\},$$

$$P^+(\kappa, j, \bar{X}) := \{x \in \bar{X} | \ell_{\kappa_i i} \leq x_i \leq u_{\kappa_i i}, i = 1, \ldots, n_1; x_j \geq u_{\kappa_j j}\}.$$

We also define $\mathcal{H}_j^\kappa(\bar{X}) := \mathrm{clconv}(P^-(\kappa, j, \bar{X}) \cup P^+(\kappa, j, \bar{X}) \setminus \emptyset)$, where empty sets are removed from consideration, and clconv denotes the closure of the convex hull.

THEOREM 1. *Assume that the set X as defined in* (1) *is non-empty and has bounded integer variables. Then for any unit partition $\mathcal{P}^*$,*

$$\mathrm{clconv}(X) = \mathrm{clconv}\{\cup_{\kappa \in K(\mathcal{P}^*)}[\mathcal{H}_{n_1}^\kappa(\mathcal{H}_{n_1-1}^\kappa(\cdots(\mathcal{H}_1^\kappa(X_L))\cdots)) \setminus \emptyset]\}.$$

Proof. The set $K(\mathcal{P}^*)$ decomposes the problem into boxes of at most unit size each of which can be sequentially convexified. To see this, note that for a given κ, the facial disjunctive description of the set $\mathcal{H}_{n_1}^\kappa(\mathcal{H}_{n_1-1}^\kappa(\cdots(\mathcal{H}_1^\kappa(X_L))\cdots))$, where $\ell_{\kappa_j j} \leq x_j \leq \ell_{\kappa_j j} + 1$, for all $j = 1, \ldots, n_1$, can be obtained by sequential convexification of mixed 0-1 programs by letting $\bar{x}_j = x_j - \ell_{\kappa_j j}$, $\bar{x}_j \in \{0, 1\}$ (Balas et al. 1993, Sherali and Adams 1994, Lovász and Schrijver 1991). As the extreme points of these polyhedra have binary values for $\bar{x}_j$, and $\ell_{\kappa_j j}$ are integer, $j = 1, \ldots, n_1$, the polyhedron given by the union of all such polyhedra have integer values for x_j in its extreme points, and hence the result follows. $\square$

Note that the introduction of binary variables in the proof of Theorem 1 is done only for expositional clarity. If one does not introduce the binary variables, the proof follows from sequential convexification of facial disjunctive programs.

EXAMPLE 1. **OM01.** Owen and Mehrotra (2001) give an example illustrating that a rudimentary cutting plane algorithm using elementary disjunctions may be infinitely convergent, where

$$X = \left\{ x \in \mathbb{Z}^2 \,\middle|\, \begin{array}{c} 8x_1 + 12x_2 \leq 27 \\ 8x_1 + 3x_2 \leq 18 \\ 0 \leq x_1, x_2 \leq 3 \end{array} \right\}, \quad c = (-1, -1).$$

The feasible region is illustrated in Figure 1. For this example, a unit partition $\mathcal{P}^*$ is given by

$x_j \in \{[0,1], [1,2], [2,3]\}$ for $j = 1, 2$. Thus $t_j = 3$ and $\kappa_j \in \{1, 2, 3\}$ for $j = 1, 2$. Therefore,

$$K(\mathcal{P}^*) = \{(1,1), (1,2), (1,3), (2,1), (2,2), (2,3), (3,1), (3,2), (3,3)\}.$$
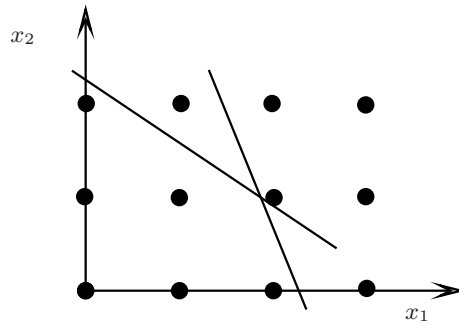

Figure 1   OM01: Feasible set.

Consider each $\kappa \in K(\mathcal{P}^*)$. For $\kappa = (1,1)$, $\mathcal{H}_2^{(1,1)}(\mathcal{H}_1^{(1,1)}(X_L))$ gives the convex hull of the

points $(0,0),(0,1)$, $(1,0)$ and $(1,1)$; for $\kappa = (1,2)$, $\mathcal{H}_2^{(1,2)}(\mathcal{H}_1^{(1,2)}(X_L))$ gives the convex hull of

the points $(0,2),(0,1),(1,1)$; for $\kappa = (2,1)$, $\mathcal{H}_2^{(2,1)}(\mathcal{H}_1^{(2,1)}(X_L))$ gives the convex hull of the

points $(1,0),(2,0),(1,1)$. For $\kappa = (1,3)$, $\mathcal{H}_2^{(1,3)}(\mathcal{H}_1^{(1,3)}(X_L))$ gives the point $(0,2)$; for $\kappa = (2,2)$,

$\mathcal{H}_2^{(2,2)}(\mathcal{H}_1^{(2,2)}(X_L))$ gives the point $(1,1)$; and for $\kappa = (3,1)$, $\mathcal{H}_2^{(3,1)}(\mathcal{H}_1^{(3,1)}(X_L))$ gives the point $(2,0)$.

For all other $\kappa \in K(\mathcal{P}^*)$, $\mathcal{H}_2^{\kappa}(\mathcal{H}_1^{\kappa}(X_L)) = \emptyset$. Taking the union of the non-empty polytopes, we get

clconv(X).

In the next section, we address the question "Is there a constructive methodology to obtain an

optimal solution to a general MILP with bounded integer variables using a disjunctive programming

characterization of its convex hull?"

3. Convex Hull Tree Algorithm

The convex hull tree is a new concept in which we construct a polyhedron X_Z such that the solution

to the problem

$$\min_{x \in X_Z} c^T x$$

yields an optimal solution to the original MILP. The construction of X_Z relies on a tree that guides the convexification process. We propose an algorithm to construct X_Z, which we refer to as the *convex hull tree algorithm*, because it is similar to a branch-and-bound tree, but instead of searching the tree, we form partial convex hulls based on the tree. The pseudo-code of the proposed algorithm is given in Algorithm 1.

At iteration k, an index i represents a distinct subset, Q_i, in which a solution could fall. As in branch-and-bound methods, each subset Q_i is a cross product of n_1 intervals corresponding to the bounded integer variables, and $n - n_1$ half lines corresponding to the non-negative continuous variables. The collection of all such subsets forms a set $\mathcal{Q}_k$. If $x^k \in Q_i$ and $x_j^k \notin \mathbb{Z}_+$ for some $j = 1, \ldots, n_1$, then we replace Q_i with two smaller non-overlapping subsets P_k^- and P_k^+, the union of which includes all integers in Q_i (see line 4 of Algorithm 1). Whenever we encounter a subset Q_i for which $u_{ij} - \ell_{ij} \leq 1$ for $j = 1, \ldots, n_1$, we replace Q_i with the closure of the convex hull of feasible points of $Q_i \cap X_k \cap \mathbb{Z}^{n_1}$, using sequential convexification of facial disjunctive programs, where for ease of notation, $X_k \cap \mathbb{Z}^{n_1}$ refers to $X_k \cap (\mathbb{Z}^{n_1} \times \mathbb{R}^{n-n_1})$, throughout the paper. At the end of iteration k, the collection of subsets is updated as $\mathcal{Q}_{k+1}$. We form a new set $X_{k+1} := \text{clconv}\{\cup_{Q_t \in \mathcal{Q}_{k+1}}(Q_t \cap X_k)\}$. We stop when we obtain an integral optimal solution to $x^k \in \arg\min_{x \in X_k} c^T x$ and let $X_Z = X_k$ at termination. Even though there is an underlying tree in the convex hull tree algorithm, we do not introduce a tree notation in this section to ease the exposition.

PROPOSITION 1. *Assume that the set X as defined in* (1) *is non-empty and has bounded integer variables. Then the convex hull tree algorithm constructs X_Z in finitely many iterations.*

Proof. Since the MILP has bounded integers, the splitting invoked in line 4 will, in finite time, yield a unit partition in the worst case, which will lead to the execution of line 7, in finite time. From Theorem 1 it follows that the convex hull tree algorithm constructs X_Z in finitely many iterations. $\square$

Example 1 **OM01.** (cont.) We illustrate each iteration of the convex hull tree algorithm.

Algorithm 1 Convex hull tree algorithm

1: Initialization: $k = 1$, $\mathcal{Q}_k := \{Q_1 = \times_{j=1}^{n_1}[0, u_j] \times \mathbb{R}_+^{n-n_1}\}$. Let $X_k = X_L \cap Q_1$, and $x^k \in$

$\arg\min_{x \in X_k} c^T x$. In case of multiple optima, let x^k be a vertex of X_k.

2: **while** $X_k \neq \emptyset$ and $x_j^k \notin \mathbb{Z}_+, j = 1, \ldots, n_1$ **do**

3:     Choose a variable $x_j^k \notin \mathbb{Z}$. Find the (unique) subset $\bar{Q}$ in the list $\mathcal{Q}_k$ such that $x^k \in \bar{Q}$. Remove

the set $\bar{Q}$ from $\mathcal{Q}_k$ and denote the revised list by $\bar{\mathcal{Q}}_k$.

4:     Form two subsets $P_k^- = \{x \in \bar{Q} | x_j \leq \lfloor x_j^k \rfloor\}$ and $P_k^+ = \{x \in \bar{Q} | x_j \geq \lceil x_j^k \rceil\}$. Let $P_k^- = \emptyset$ (or

$P_k^+ = \emptyset$) if $P_k^- \cap X_k = \emptyset$ (or $P_k^+ \cap X_k = \emptyset$). Define the new list by $\mathcal{Q}_{k+1} = \bar{\mathcal{Q}}_k \cup P_k^- \cup P_k^+ \setminus \emptyset$

and denote all subsets in the updated list $\mathcal{Q}_{k+1}$ by $\{Q_t\}_{t=1}^T$.

5:     **for** $t = 1, \ldots, T$ **do**

6:         **if** $u_{ti} - \ell_{ti} \leq 1$ for $i = 1, \ldots, n_1$ **then**

7:             Let $Q_t \leftarrow \text{clconv}(Q_t \cap X_k \cap \mathbb{Z}^{n_1})$.

8:         **end if**

9:     **end for**

10:     Form the set $X_{k+1} := \text{clconv}\{\cup_{t=1}^T (Q_t \cap X_k)\}$.

11:     Let $k \leftarrow k + 1$. Find $x^k \in \arg\min_{x \in X_k} c^T x$ such that x^k is a vertex of X_k.

12: **end while**

13: Return $X_Z = X_k$.

Iteration 1 Initially, $\mathcal{Q}_1 := Q_1 = [0, 3] \times [0, 3]$, $X_1 = X_L$ and $x^1 = (15/8, 1)$. Because $x_1^1 \notin \mathbb{Z}$ and

$x^1 \in Q_1$, we replace the set Q_1 with the two sets $P_1^- = \{x \in Q_1 | x_1 \leq \lfloor \frac{15}{8} \rfloor\}$ and $P_1^+ = \{x \in Q_1 | x_1 \geq$

$\lceil \frac{15}{8} \rceil\}$. The updated list is $\mathcal{Q}_2 := \{Q_1 = [0, 1] \times [0, 3], Q_2 = [2, 3] \times [0, 3]\}$. Hence, we have $X_2 =$

$\text{clconv}[(X_1 \cap Q_1) \cup (X_1 \cap Q_2)]$. The facet of X_2 generated in line 10 passes through the points

$(1, 19/12)$ and $(2, 2/3)$, which deletes x^1. This is the same cut as that of Owen and Mehrotra (2001).

The next point obtained at the end of this iteration is $x^2 = (2, 2/3)$.

Iteration 2 Because $x_2^2 \notin \mathbb{Z}$ and $x^2 \in Q_2$, we replace the set Q_2 with the two sets $P_2^- = \{x \in Q_2 | x_2 \leq$

$\lfloor \frac{2}{3} \rfloor\}$ and $P_2^+ = \{x \in Q_2 | x_2 \geq \lceil \frac{2}{3} \rceil\}$. The updated subsets are $Q_1 = [0, 1] \times [0, 3], Q_2 = [2, 3] \times \{0\}, Q_3 =$

$[2,3] \times [1,3]$. Note that $X_2 \cap Q_3$ is infeasible, so Q_3 is removed from consideration. In line 7 of Iteration 2, because $2 \le x_1 \le 3, x_2 = 0$, we convexify the set $\{x \in X_2 \cap Q_2 \cap \mathbb{Z}^2\}$, which results in a single feasible point $(2,0)$, and we update the set $Q_2 \leftarrow \{2\} \times \{0\}$. Therefore, $\mathcal{Q}_3 := \{Q_1 = [0,1] \times [0,3], Q_2 = \{2\} \times \{0\}\}$. Hence, we have $X_3 = \text{clconv}[(X_2 \cap Q_1) \cup (X_2 \cap Q_2)]$. The facet of X_3 generated in line 10 goes through $(1,19/12)$ and $(2,0)$, which deletes x^2. As a result, we get a deeper cut than that obtained from the procedure of Owen and Mehrotra (2001). The next point obtained at the end of this iteration is $x^3 = (1, 19/12)$.

Iteration 3 Because $x_2^3 \notin \mathbb{Z}$ and $x^3 \in Q_1$, we replace the set Q_1 with the two sets $P_3^- = \{x \in Q_1 | x_2 \le \lfloor \frac{19}{12} \rfloor\}$ and $P_3^+ = \{x \in Q_1 | x_2 \ge \lceil \frac{19}{12} \rceil\}$. The updated list is $\mathcal{Q}_4 := \{Q_1 = [0,1] \times [0,1], Q_2 = \{2\} \times \{0\}, Q_3 = [0,1] \times [2,3]\}$. Similar to Iteration 2, we convexify the set $\{x \in X_3 \cap Q_3 \cap \mathbb{Z}^2\}$, which results in a single feasible point $(0,2)$, and we update the set $Q_3 \leftarrow \{0\} \times \{2\}$. We also convexify the set $\{x \in X_3 \cap Q_1 \cap \mathbb{Z}^2\}$, which gives Q_1 itself. Hence, we have $X_4 = \text{clconv}[(X_3 \cap Q_1) \cup (X_3 \cap Q_2) \cup (X_3 \cap Q_3)]$. The facet of X_4 generated in line 10 goes through $(0,2)$ and $(2,0)$ and thus gives the convex hull of X, which deletes x^3.

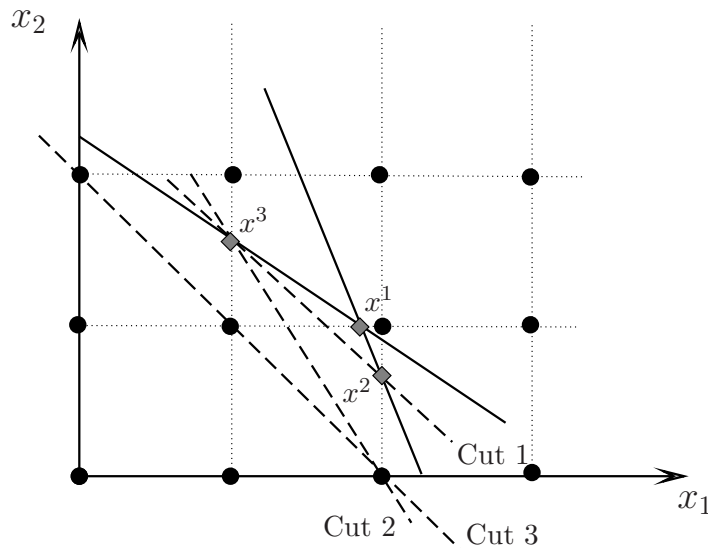We illustrate these iterations in Figure 2.

Figure 2   OM01: Convex hull tree cuts.

4. Cutting plane tree algorithm

In this section, we address the question: "If we are restricted to introduce only one cutting plane in any iteration, is there a finitely convergent disjunctive programming algorithm that solves a general MILP?" In other words, we consider a simplification of Algorithm 1, which we refer to as the cutting plane tree algorithm.

In the cutting plane tree $\mathcal{T}$, there is a single root node o. For each node $\sigma \in \mathcal{T}$, an integer m_σ keeps track of the cutting planes that will be used to generate a disjunctive cut when this node is revisited, an integer $v_\sigma \in \{1, 2, \ldots, n_1\}$ stores the index of the integer variable that is split, an integer q_σ stores the (lower) level of the splitting. Let l_σ, r_σ and p_σ denote links to the left child, right child and parent nodes of node σ, respectively. Let $\mathcal{S}(\sigma)$ be all nodes on the subtree rooted at node σ (not including node σ and the leaf nodes). Let $\mathcal{N}(\sigma)$ be the collection of the nodes on the path from the root node to node σ (not including the root node), let $\mathcal{N}^-(\sigma)$ be the collection of nodes in $\mathcal{N}(\sigma)$ that were formed as the left child node of its parent, and let $\mathcal{N}^+(\sigma)$ be the collection of nodes in $\mathcal{N}(\sigma)$ that were formes as the right child node of its parent. Given $\sigma \in \mathcal{T}$ define

$$\mathcal{C}_\sigma = \{x | x_j \in [0, u_j], x_{v_{p_s}} \leq q_{p_s}, \forall s \in \mathcal{N}^-(\sigma), x_{v_{p_s}} \geq q_{p_s} + 1, \forall s \in \mathcal{N}^+(\sigma)\}.$$

We let m_σ store an iteration index, which gives the set X_{m_σ} to be used in the cut generation LP (CGLP) (Balas 1979, Sherali and Shetty 1980). The set X_{m_σ} corresponds to X_L together with the first $m_\sigma - 1$ cuts added to it. If $X_{m_\sigma} \cap \mathcal{C}_{l_\sigma} = \emptyset$ ($X_{m_\sigma} \cap \mathcal{C}_{r_\sigma} = \emptyset$), we say that the left (right) child node of σ is "fathomed", i.e., $l_\sigma = null$ ($r_\sigma = null$).

Now that we have introduced the tree notation necessary in the description of the cutting plane tree algorithm, we relate the notation in this section to the notation introduced in Sections 2 and 3. In Algorithm 2, $\mathcal{L}_{k+1}$ denotes the collection of all leaf nodes of the cutting plane tree at the end of iteration k. Note that, each feasible region given by $\mathcal{C}_\sigma$ for $\sigma \in \mathcal{L}_{k+1}$ defines a subset Q_i in iteration k. The collection of these non-overlapping subsets Q_i, for $i = 1, \ldots, |\mathcal{L}_{k+1}|$ at iteration k, denoted by $\mathcal{Q}_{k+1}$, gives a valid partition, $\mathcal{P}$ of $\times_{j=1}^{n_1}[0, u_j] \times \mathbb{R}_+^{n-n_1}$.

The pseudo-code of the cutting plane tree algorithm is given in Algorithm 2. At iteration k, if the current extreme point solution to $\min_{x \in X_k} c^T x$, given by x^k is integral, then we have found the optimal solution to the MILP. Otherwise, we search the cutting plane tree, to find the last node σ on the path from the root node such that $x^k \in \mathcal{C}_\sigma$. There are two cases: Case (1) σ is a leaf node ($\sigma \in \mathcal{L}_k$), Case (2) σ is not a leaf node ($\sigma \notin \mathcal{L}_k$, $x^k \in \mathcal{C}_\sigma$, $x^k \notin \mathcal{C}_{l_\sigma}$ and $x^k \notin \mathcal{C}_{r_\sigma}$). In Case (1), we choose a fractional variable x_j, $j = 1, \ldots, n_1$ with the smallest index, and let the split variable be $v_\sigma = j$. We create two new nodes: left (l_σ) and right (r_σ) children of σ at the split level $q_\sigma = \lfloor x_j \rfloor$. We let $\mathcal{C}_{l_\sigma} = \{x \in \mathcal{C}_\sigma | x_j \leq \lfloor x_j^k \rfloor\}$ and $\mathcal{C}_{r_\sigma} = \{x \in \mathcal{C}_\sigma | x_j \geq \lceil x_j^k \rceil\}$. In this case, we also let $m_\sigma = k$, as this is the first time the tree search for a fractional solution stops at node σ. In Case (2), the cutting plane tree and m_σ are unchanged. However, in this case, we update $m_t = k$ for all successors of σ, $t \in \mathcal{S}(\sigma)$. We generate a valid inequality for the set clconv$\{\cup_{t \in \mathcal{L}_{k+1}}(X_{m_\sigma} \cap \mathcal{C}_t)\}$ that cuts off x^k (from an extreme point of the CGLP). The new inequality is included along with those defining X_k, and the resulting set is denoted X_{k+1}. This process continues until one of the stopping criteria is satisfied.

PROPOSITION 2. *Assume that the set X as defined in* (1) *is non-empty and has bounded integer variables. Then the cutting plane tree algorithm converges to the optimal solution in finitely many iterations.*

Proof. Note that the cutting plane tree cannot expand infinitely. In the worst case, the leaf nodes of the tree form a unit partition $\mathcal{P}^*$. Therefore, the number of leaf nodes is no more than $\Pi_{j=1}^{n_1}(1 + u_j)$, where u_j is the integer upper bound of variable $x_j, j = 1, \ldots, n_1$. Let σ be the node found in line 3 of Algorithm 2 at iteration k. There are two cases: (1) $\sigma \in \mathcal{L}_k$ and $x^k \in \mathcal{C}_\sigma$, or (2) $\sigma \notin \mathcal{L}_k$, $x^k \in \mathcal{C}_\sigma$, $x^k \notin \mathcal{C}_{l_\sigma}$ and $x^k \notin \mathcal{C}_{r_\sigma}$. In Case (1), the cutting plane tree is expanded by splitting from node σ, and we update the collection of leaf nodes as $\mathcal{L}_{k+1}$. As the number of possible leaf nodes is finite, this case can happen only finitely many times. In Case (2), the tree remains constant and we have $\mathcal{L}_{k+1} = \mathcal{L}_k$. In either case, a new extreme point of the CGLP is generated. As there are finitely many extreme points of the CGLP, and a subset of these corresponds to all facets of

---

Algorithm 2 Cutting plane tree algorithm

1: Initialization: $k = 1$, create a node o, where $p_o = null$, $m_o = 1$. Let $\mathcal{T} = \{o\}$, $\mathcal{L}_k := \{o\}$, $\mathcal{C}_o = \{x | x_j \in [0, u_j], j = 1, \ldots, n_1\}$, $X_k = X_L$ and $x^k \in \arg\min_{x \in X_k} c^T x$, where x^k is a vertex of X_k.

2: **while** $X_k \neq \emptyset$ and $x_j^k \notin \mathbb{Z}_+, j = 1, \ldots, n_1$ **do**

3:     Search for node $\sigma \in \mathcal{T}$ such that: either $\sigma \in \mathcal{L}_k$ and $x^k \in \mathcal{C}_\sigma$; or, node $\sigma \notin \mathcal{L}_k$ and $x^k \in \mathcal{C}_\sigma$, $x^k \notin \mathcal{C}_{l_\sigma}$ and $x^k \notin \mathcal{C}_{r_\sigma}$.

4:     **if** $\sigma \in \mathcal{L}_k$ **then**

5:         Choose the smallest index j such that variable $x_j^k \notin \mathbb{Z}$. Let $m_\sigma = k$. Let $\mathcal{L}_{k+1} \leftarrow \mathcal{L}_k \setminus \{\sigma\}$.

6:         Create a node l^- with $l_\sigma = l^-$, $p_{l^-} = \sigma$ and a node l^+ with $r_\sigma = l^+$, $p_{l^+} = \sigma$.

7:         Let $\mathcal{C}_{l^-} = \{x \in \mathcal{C}_\sigma | x_j \leq \lfloor x_j^k \rfloor\}$ and $\mathcal{C}_{l^+} = \{x \in \mathcal{C}_\sigma | x_j \geq \lceil x_j^k \rceil\}$.

8:         **if** $\mathcal{C}_{l^-} \cap X_k \neq \emptyset$ **then**

9:             Let $\mathcal{L}_{k+1} \leftarrow \mathcal{L}_{k+1} \cup \{l^-\}$. Let $\mathcal{T} \leftarrow \mathcal{T} \cup \{l^-\}$.

10:         **else**

11:             Let $l^- = null$ (fathom).

12:         **end if**

13:         **if** $\mathcal{C}_{l^+} \cap X_k \neq \emptyset$ **then**

14:             Let $\mathcal{L}_{k+1} \leftarrow \mathcal{L}_{k+1} \cup \{l^+\}$. Let $\mathcal{T} \leftarrow \mathcal{T} \cup \{l^+\}$.

15:         **else**

16:             Let $l^+ = null$ (fathom).

17:         **end if**

18:         Let $v_\sigma = j$, $q_\sigma = \lfloor x_j^k \rfloor$.

19:     **else if** $\sigma \notin \mathcal{L}_k$ and $x^k \in \mathcal{C}_\sigma$, $x^k \notin \mathcal{C}_{l_\sigma}$ and $x^k \notin \mathcal{C}_{r_\sigma}$ **then**

20:         Let $\mathcal{L}_{k+1} \leftarrow \mathcal{L}_k$, let $m_t \leftarrow k, \forall t \in \mathcal{S}(\sigma)$.

21:     **end if**

22:     Generate a valid inequality for the set $\text{clconv}\{\cup_{t \in \mathcal{L}_{k+1}}(X_{m_\sigma} \cap \mathcal{C}_t)\}$ that cuts off x^k (using an extreme point of the CGLP). Call the set with the additional valid inequality X_{k+1}.

23:     Let $k \leftarrow k + 1$ and $x^k \in \arg\min_{x \in X_k} c^T x$, where x^k is a vertex of X_k.

24: **end while**

---

clconv$\{\cup_{t\in\mathcal{L}_{k+1}}(X_{m_\sigma}\cap\mathcal{C}_t)\}$ we will have either $x_{v_\sigma}\leq q_\sigma$ or $x_{v_\sigma}\geq q_\sigma+1$ in finitely many iterations. In either case, the tree search in Line 3 stops at l_σ, r_σ or one of its successors. As a result, the algorithm can visit node σ only finitely many times. Consequently, there exists a finite number, N, such that either the algorithm stops before iteration N, or at iteration N, a unit partition $\mathcal{P}^*$ is found.

If we reach a point where the leaf nodes correspond to a unit partition $\mathcal{P}^*$, then each path from the root node of $\mathcal{T}$ to a leaf node corresponds to a vector $\kappa\in K(\mathcal{P}^*)$ as defined in Section 2. The order in which the variables are split on this path defines a sequence. Note that the variable x_j can be split more than once, in which case we consider the last node it was split, denoted by σ_j. Without loss of generality, assume that this sequence of variables is given by $1,\ldots,n_1$ so that $\sigma_j\in\mathcal{S}(\sigma_i)$ for $i<j$. As a result, when a node σ_i is visited and a cut is added at this node, we update m_{σ_j} to include this cut for all $\sigma_j\in\mathcal{S}(\sigma_i)$. Therefore, for a given κ, we sequentially convexify $X_{m_{\sigma_1}}$ with respect to x_1 first. We convexify the resulting set with respect to x_2 next, and continue until we reach variable x_{n_1}. The union of the nonempty sets defined by each leaf node (or each $\kappa\in K(\mathcal{P}^*)$) gives us clconv$(X)$ from Theorem 1. Therefore, Algorithm 2 converges to the optimal solution in finitely many iterations. $\quad\square$

The reader might find it interesting to compare the concepts introduced in this paper with another recent paper by Jörg (2007) who also addresses the finiteness of disjunctive programming for general MILP. His approach proposes the use of cuts obtained from multiple split disjunctions in the projected space of integer variables. The author proved that for bounded MILP, there exists a conjunction of n_1+1 split disjunctions that characterizes the mixed-integer hull of the MILP. The author devises a two phase algorithm as follows: Phase 1) solve a relaxation, and ascertain whether the optimal face includes any integer points. If an integer optimum is identified then the method stops. Otherwise, the fractional optimum point is deleted using a Gomory cut. This phase continues until the final fractional alternative optimum is identified as x^*. This point is deleted in the projected space, which constitutes the second phase: Phase 2) find all extreme directions of a polyhedral cone to reduce the inequalities to the space of the integer variables. Now using

a conjunction of multiple split disjunctions one can delete the projection of x^* in the projected space of integer variables. The process then repeats from phase 1. The author proves that this process is finite. However, the identification of all alternative optima (in phase 1), as well as the identification of all extreme directions (in phase 2) suggest that the algorithmic map producing the subsequence of polyhedra (observed after successive phase 2 iterations) involves the solution of multiple NP-hard problems.

5. Examples from the Literature

In this section, we illustrate finite convergence of the cutting plane tree algorithm for three examples from the literature (Cook et al. 1990, Owen and Mehrotra 2001, Sen and Sherali 1985). Each of these examples illustrates specific properties of the cutting plane tree algorithm and to the best of our knowledge, most of the examples have defied finite convergence for various approaches based on linear disjunctions.

Example 1 **OM01.** (cont.) This example illustrates that finite convergence of the cutting plane tree algorithm can be obtained even when the facial disjunctive property is absent. We observe that using the cutting plane tree algorithm, OM01 is solved in 7 iterations. Table 1 illustrates each iteration of Algorithm 2. Table 1 also provides the node σ that is visited and the value of m_σ for each iteration. Note that Case (1) of the proof of Proposition 2 applies at each iteration. Figure 3 depicts the first and the last polyhedra. We refer the reader to Figure 6 in Appendix 5 for an illustration of the cutting plane tree at termination.

A similar example without previously known finite convergence of disjunctive cuts, appears in Figure 2 of Sen and Sherali (1985). Since the behavior of the cutting plane tree algorithm for that instance is similar to the above illustration, we omit it from the present discussion.

EXAMPLE 2. **CKS90.** Unlike the pure integer linear program in Example 1, Cook et al. (1990) provide an MILP example which illustrates that a cutting plane procedure based on split cuts could take infinitely many iterations. In contrast to split or elementary disjunctions, Algorithm 2 works on multi-term disjunctions, which when guided by the tree, leads to a finitely convergent algorithm

Table 1 OM01 example

k	x^k	σ	m_σ	$\mathcal{Q}_{k+1}$	cut
1	$(15/8, 1)$	1	1	$Q_1 = [0,1] \times [0,3]$ $Q_2 = [2,3] \times [0,3]$	$11/12 x_1 + x_2 \le 5/2$
2	$(2, 2/3)$	3	2	$Q_1 = [0,1] \times [0,3]$ $Q_2 = [2,3] \times [0,0]$	$x_1 + 15/19 x_2 \le 9/4$
3	$(1, 19/12)$	2	3	$Q_1 = [2,3] \times [0,0]$ $Q_2 = [0,1] \times [0,1]$ $Q_3 = [0,1] \times [2,3]$	$x_1 + 15/16 x_2 \le 9/4$
4	$(3/8, 2)$	6	4	$Q_1 = [2,3] \times [0,0]$ $Q_2 = [0,1] \times [0,1]$ $Q_3 = [0,0] \times [2,3]$	$x_1 + x_2 \le 9/4$
5	$(9/4, 0)$	4	5	$Q_1 = [0,1] \times [0,1]$ $Q_2 = [0,0] \times [2,3]$ $Q_3 = [2,2] \times [0,0]$	$9 x_1 + 8 x_2 \le 18$
6	$(0, 9/4)$	7	6	$Q_1 = [0,1] \times [0,1]$ $Q_2 = [2,2] \times [0,0]$ $Q_3 = [0,0] \times [2,2]$	$x_1 + x_2 \le 2$
7	$(2, 0)$				
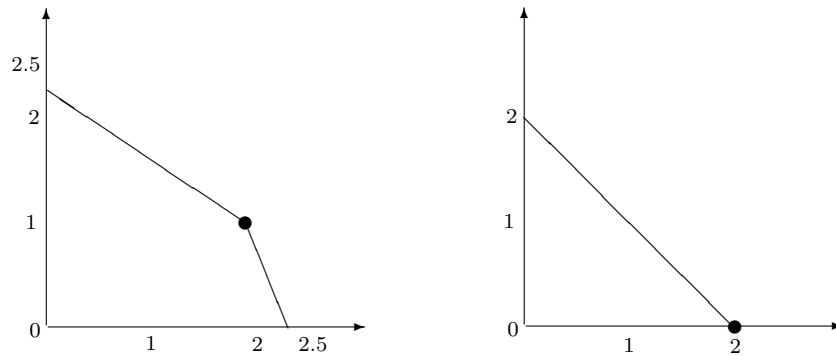
Figure 3 OM01: the first and last X_k.

that provides an integral solution. For this example, Andersen et al. (2007) use inequalities derived from two rows of the simplex tableau and Li and Richard (2008) use inequalities derived from a nonlinear (multiplicative) disjunction to get the optimal solution in finite steps. In this example:

$$X = \left\{ x \in \mathbb{Z}^2 \times \mathbb{R}_+ \Big| \begin{array}{l} x_1 - x_3 \ge 0 \\ x_2 - x_3 \ge 0 \\ x_1 + x_2 + 2x_3 \le 2 \\ 0 \le x_1, x_2 \le 2 \end{array} \right\}, \ c = (0, 0, -1).$$

Table 2 provides a summary of each iteration of Algorithm 2. Figure 4 depicts the first and the last polyhedra. Figure 7 in Appendix 5 provides a more detailed illustration of the cutting plane tree at each iteration.

EXAMPLE 3. **SS85.** This example appears as Figure 1 in Sen and Sherali (1985) where they show

Table 2   CKS90 example

k	x^k	σ	m_σ	$\mathcal{Q}_{k+1}$	cut
1	$(1/2, 1/2, 1/2)$	1	1	$Q_1 = [0,0] \times [0,2] \times \mathbb{R}_+$ $Q_2 = [1,2] \times [0,2] \times \mathbb{R}_+$	$x_1 - 3x_3 \geq 0$
2	$(1, 1/3, 1/3)$	3	2	$Q_1 = [0,0] \times [0,2] \times \mathbb{R}_+$ $Q_2 = [1,2] \times [0,0] \times \mathbb{R}_+$ $Q_3 = [1,2] \times [1,2] \times \mathbb{R}_+$	$x_3 \leq 0$
3	$(0,0,0)$				
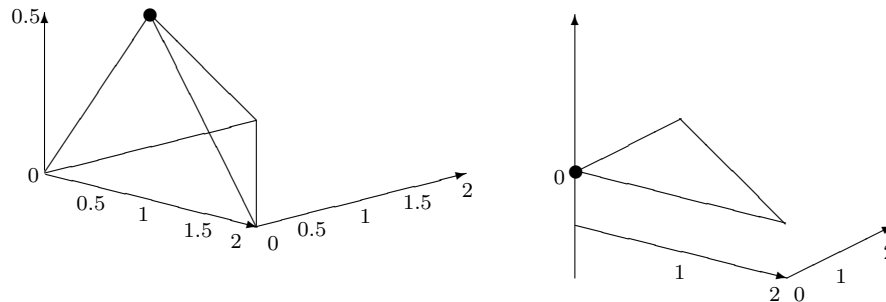
Figure 4   CKS90: the first and last X_k.

that sequential cutting plane algorithms can fail to converge when appropriate memory of cuts is not maintained. The tree in Algorithm 2 provides a convenient mechanism to record such memory (via m_σ). In this example:

$$X = \left\{ x \in \mathbb{Z}^3 \middle| \begin{array}{c} x_1 + 2x_2 - 2x_3 \geq 0 \\ 2x_1 + 2x_2 - 3x_3 \geq 0 \\ 2x_1 + x_2 - 2x_3 \geq 0 \\ 2x_1 + 2x_2 \leq 3 \\ 0 \leq x_1, x_2, x_3 \leq 1 \end{array} \right\}, \ c = (0, 0, -1).$$

Table 3 illustrates each iteration of Algorithm 2. Note that in all previous examples, at each iteration k, Case (1) in the proof of Proposition 2 applies, and so the set X_{m_σ} used in CGLP is equivalent to X_k. However, for SS85, in iteration 3, case (2) of the proof of Proposition 2 applies, and X_{m_σ} used in CGLP is different than X_k. Figure 5 depicts the first and the last polyhedra. We refer the reader to Figure 8 in Appendix 5 for an illustration of the cutting plane tree at termination.

In all previous examples, our algorithm obtains the convex hull of feasible solutions in the last polyhedron, while in general this is not always true. Note that in all of the examples, the number of nonempty subsets Q_i in any iteration is modest. This observation provides preliminary evidence that our approach can be implemented in a manner that the convexification process is guided by

Table 3   SS85 example

k	x^k	σ	m_σ	$\mathcal{Q}_{k+1}$	cut
1	$(1, 1/2, 1)$	1	1	$Q_1 = [0,1] \times [0,0] \times [0,1]$ $Q_2 = [0,1] \times [1,1] \times [0,1]$	$x_1 + 1/2x_2 \leq 1$
2	$(1/2, 1, 1)$	3	2	$Q_1 = [0,1] \times [0,0] \times [0,1]$ $Q_2 = [0,0] \times [1,1] \times [0,1]$	$x_1 + x_2 - 2x_3 \geq 0$
3	$(1/2, 1, 3/4)$	3	2	$Q_1 = [0,1] \times [0,0] \times [0,1]$ $Q_2 = [0,0] \times [1,1] \times [0,1]$	$x_1 + x_2 \leq 1$
4	$(1, 0, 1/2)$	2	4	$Q_1 = [0,0] \times [1,1] \times [0,1]$ $Q_2 = [0,1] \times [0,0] \times [0,0]$	$x_2 - 2x_3 \geq 0$
5	$(0, 1, 1/2)$	4	5	$Q_1 = [0,1] \times [0,0] \times [0,0]$ $Q_2 = [0,0] \times [1,1] \times [0,0]$	$x_3 \leq 0$
6	$(0, 0, 0)$				
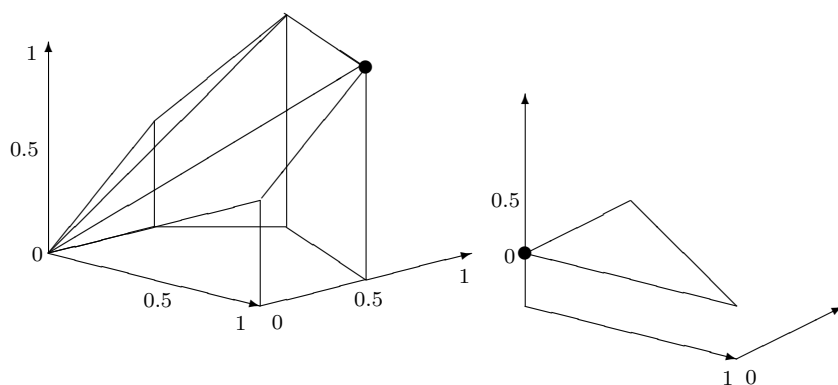
Figure 5   SS85: the first and last X_k.

the objective function, and consequently, fewer inequalities may be generated. However, for large instances, it may become necessary to design more efficient ways to solve cut generation problems using special purpose techniques in the spirit of Perregaard and Balas (2001), Balas and Perregaard (2003). It is natural to incorporate the disjunctive cuts proposed in this paper within a standard branch-and-cut methodology. Finally, our development suggests a Progressive RLT Process which adapts to sequential optimization, instead of having to choose the RLT level in the hierarchy, a priori. Work along these lines will be reported in subsequent papers.

Appendix. Cutting Plane Trees

Example 1 **OM02.** (cont.) Figure 6 depicts the cutting plane tree at termination. The blank nodes (without a number) in Figure 6 indicate nodes for which $X_k \cap C_\sigma = \emptyset$.
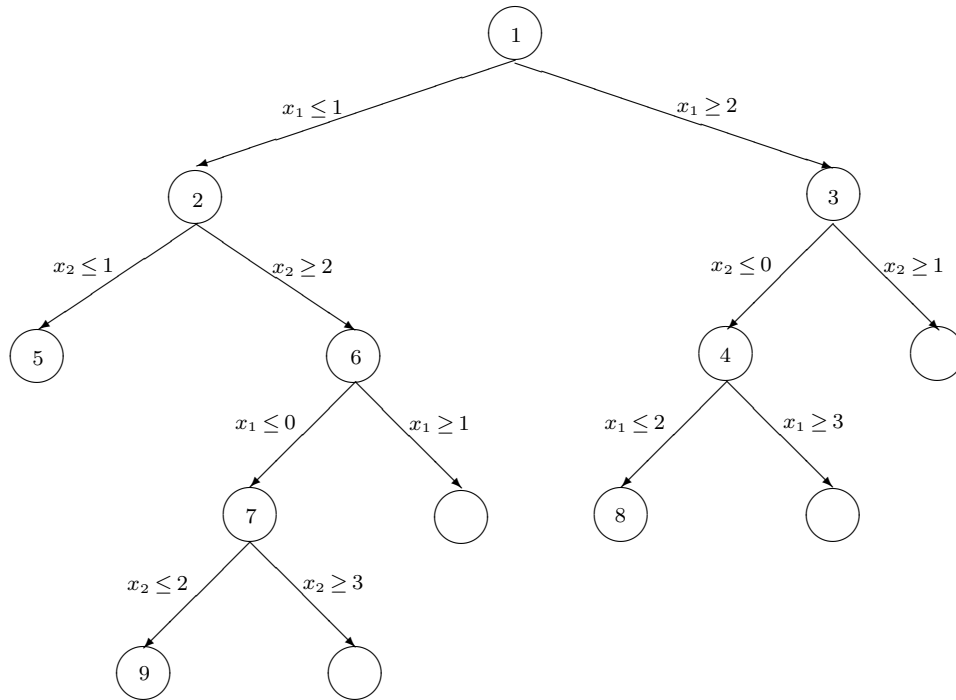
Figure 6   OM01: The final cutting plane tree.

Example 2 **CKS90.** (cont.) Because it takes two iterations to solve this example, we illustrate the cutting plane tree at each iteration in Figure 7.
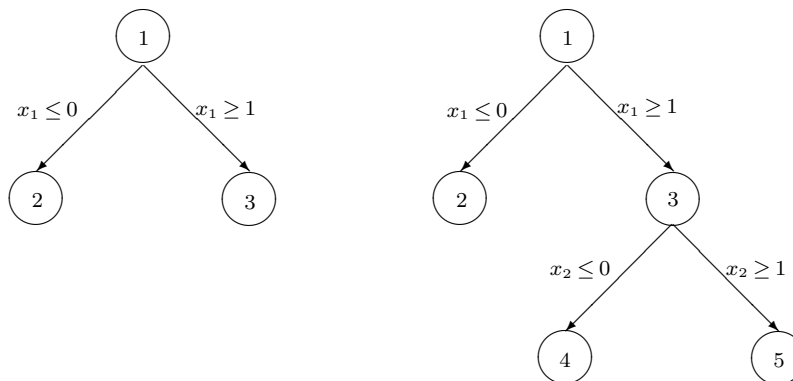

Figure 7   CKS90: The cutting plane tree of iterations 1 and 2.

Example 3 **SS85.** (cont.) Figure 8 depicts the cutting plane tree at termination.

Acknowledgments
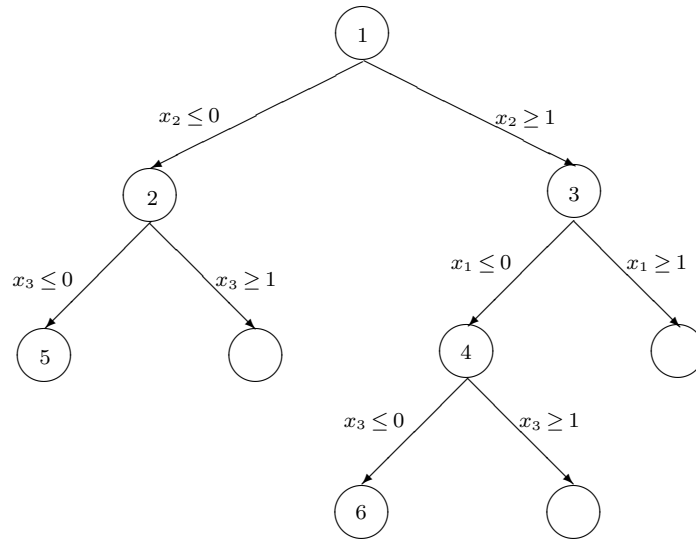
Figure 8   SS85: The final cutting plane tree.

References

Adams, W. P., H. D. Sherali. 2005. A hierarchy of relaxations leading to the convex hull representation for

general discrete optimization problems. *Annals of Operations Research* **140**(1) 21–47.

Andersen, Kent, Quentin Louveaux, Robert Weismantel, Laurence A. Wolsey. 2007. Inequalities from two

rows of a simplex tableau. *IPCO '07: Proceedings of the 12th International Conference on Integer

Programming and Combinatorial Optimization*. Springer-Verlag, Berlin, Heidelberg, 1–15. doi:http:

//dx.doi.org/10.1007/978-3-540-72792-7_1.

Balas, E. 1979. Disjunctive programming. *Annals of Discrete Mathematics* **5** 3–51.

Balas, E., S. Ceria, G. Cornuéjols. 1993. A lift-and-project cutting plane algorithm for mixed 0-1 programs.

Mathematical Programming **58**(1–3) 295–324.

Balas, E., M. Perregaard. 2003. A precise correspondence between lift-and-project cuts, simple disjunctive

cuts, and mixed integer Gomory cuts for 0-1 programming. *Mathematical Programming* **94**(2–3) 221–

245.

Bonami, P., L. T. Biegler, A. R. Conn, G. Cornuéjols, I. E. Grossmann, C. D. Laird, J. Lee, A. Lodi,

F. Margot, N. Sawaya, A. Wachter. 2008. An algorithmic framework for convex mixed integer nonlinear programs. *Discrete Optimization* **5**(2) 186–204.

Cook, W., R. Kannan, A. Schrijver. 1990. Chvátal closures for mixed integer programming problems. *Mathematical Programming* **47**(1–3) 155–174.

Cornuéjols, G. 2008. Valid inequalities for mixed integer linear programs. *Mathematical Programming* **112**(1) 3–44.

Gomory, R. E. 1963. An algorithm for integer solutions to linear programs. P. Graves, P. Wolfe, eds., *Recent Advances in Mathematical Programming*. McGraw-Hill, New York, 269–302.

Jörg, M. 2007. k-disjunctive cuts and a finite cutting plane algorithm for general mixed integer linear programs. `http://arxiv.org/PS_cache/arxiv/pdf/0707/0707.3945v1.pdf`.

Li, Y., J.-P.P Richard. 2008. Cook, Kannan and Schrijver's example revisited. *Discrete Optimization* **5**(4) 724–734.

Lovász, L., A. Schrijver. 1991. Cones of matrices and set-functions and 0-1 optimization. *SIAM Journal of Optimization* **1**(2) 166–190.

Owen, J. H., S. Mehrotra. 2001. A disjunctive cutting plane procedure for general mixed-integer linear programs. *Mathematical Programming* **89**(2) 437–448.

Owen, J. H., S. Mehrotra. 2002. On the value of binary expansions for general mixed-integer programs. *Operations Research* **50**(5) 810–819.

Padberg, M. W., T. J. van Roy, L. A. Wolsey. 1985. Valid linear inequalities for fixed charge problems. *Operations Research* **33**(4) 842–861.

Perregaard, Michael, Egon Balas. 2001. Generating cuts from multiple-term disjunctions. *Proceedings of the 8th International IPCO Conference on Integer Programming and Combinatorial Optimization*. Springer-Verlag, London, UK, 348–360.

Sen, S., H. D. Sherali. 1985. On the convergence of cutting plane algorithms for a class of nonconvex mathematical programs. *Mathematical Programming* **31**(1) 42–56.

Sherali, H. D., W. P. Adams. 1990. A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. *SIAM Journal of Discrete Mathematics* **3**(3) 411–430.

Sherali, H. D., W. P. Adams. 1994. A hierarchy of relaxations and convex hull representations for mixed-integer zero-one programming problems. *Discrete Applied Mathematics* **52**(1) 83–106.

Sherali, H. D., C. M. Shetty. 1980. *Optimization with Disjunctive Constraints*. Springer Verlag, Berlin.

Van Roy, T. J., L. A. Wolsey. 1985. Valid inequalities and separation for uncapacitated fixed charge networks. *Operations Research Letters* **4**(3) 105–112.

Yuan, Y., S. Sen. 2009. Enhanced cut generation methods for decomposition-based branch and cut for two-stage stochastic mixed-integer programs. *INFORMS Journal on Computing* **21**(3) 480–487.