A Branch-and-cut Algorithm for Integer Bilevel Linear Programs

S.T. DeNegre and T.K. Ralphs

Abstract We describe a rudimentary branch-and-cut algorithm for solving integer bilevel linear programs that extends existing techniques for standard integer linear programs to this very challenging computational setting. The algorithm improves on the branch-and-bound algorithm of Moore and Bard in that it uses cutting plane techniques to produce improved bounds, does not require specialized branching strategies, and can be implemented in a straightforward way using only linear solvers. An implementation built using software components available in the COIN-OR software repository is described and preliminary computational results presented.

Key words: Bilevel Programming, Integer Programming, Branch and Cut, Valid Inequality, Branch and Bound

1 Introduction

Standard mathematical programs consider decision problems in which there is a single decision-maker (DM) controlling all variables. Many real-world decision problems involve multiple, independent DMs, whose interests are not necessarily aligned. In this paper, we discuss solution methods for a class of models known as *integer bilevel linear programs* (IBLPs) that generalize standard integer linear programming (ILP) models by considering two sets of variables, each controlled by a separate DM.

S.T. DeNegre
Department of Industrial & Systems Engineering, Lehigh University, 200 W. Packer Avenue, Bethlehem, PA 18015, e-mail: sdenegre@lehigh.edu

T.K. Ralphs
Department of Industrial & Systems Engineering, Lehigh University, 200 W. Packer Avenue, Bethlehem, PA 18015, e-mail: ted@lehigh.edu

J.W. Chinneck et al. (eds.), *Operations Research and Cyber-Infrastructure*, Operations Research/Computer Science Interfaces Series 47, DOI: 10.1007/978-0-387-88843-9_4,

The goal of the work described herein is to demonstrate that it is possible, in principle, to generalize the tremendously successful branch-and-cut framework commonly used to solve mixed integer linear programs to this very challenging computational setting. By developing techniques for IBLP that are analogous to those used in the ILP setting, we have been able to leverage the many advances that have occurred in solution technology for ILPs. Although our implementation is quite rudimentary and is intended only as a demonstration of concept, the algorithm improves on the branch-and-bound algorithm previously proposed by Moore and Bard (1990) in that it uses a basic cutting plane procedure to produce improved bounds, does not require specialized branching techniques, and can be implemented in a straightforward way using existing software.

Conceptually, the decisions in an IBLP are made in sequential order according to an implicit hierarchy. Top-level decisions are made first, after which the lower-level decisions are made under the mandates of those upper-level decisions. Formally, let $x \in \mathbb{R}^{n_1}$ represent a set of variables controlled by an *upper-level DM* or *leader* and let $y \in \mathbb{R}^{n_2}$ represent a set of variables controlled by a *lower-level DM* or *follower*. The canonical integer bilevel linear program is then given by

$$z_{IBLP} = \max\left\{c^1 x + d^1 y \mid x \in \mathscr{P}_U \cap \mathbb{Z}^{n_1}, y \in \operatorname{argmax}\{d^2 y \mid y \in \mathscr{P}_L(x) \cap \mathbb{Z}^{n_2}\}\right\},$$

where

$$\mathscr{P}_U = \left\{x \in \mathbb{R}^{n_1} \mid A^1 x \leq b^1, x \geq 0\right\}$$

is the polyhedron defining the *upper-level feasible region*;

$$\mathscr{P}_L(x) = \left\{y \in \mathbb{R}^{n_2} \mid G^2 y \leq b^2 - A^2 x, y \geq 0\right\}$$

is the polyhedron defining the *lower-level feasible region* with respect to a given $x \in \mathbb{R}^{m_1}$; $A^1 \in \mathbb{Q}^{m_1 \times n_1}$; $b^1 \in \mathbb{Q}^{m_1}$; $A^2 \in \mathbb{Q}^{m_2 \times n_1}$, $G^2 \in \mathbb{Q}^{m_2 \times n_2}$; and $b^2 \in \mathbb{Q}^{m_2}$. The defining characteristic of a bilevel program, in contrast with a standard mathematical program, is that the lower-level variables are required to consist of an optimal solution to an ILP whose right-hand side depends on the values chosen for the upper-level variables.

Bilevel models arise naturally in systems involving two opposing parties, such as in military and law enforcement applications. The essence of what makes these models difficult to analyze is the implicit adversarial relationship between the upper- and lower-level DMs stemming from the fact that improvements to the upper-level DM's objective usually come at the expense of a degradation in the lower-level DM's objective. In fact, without such an adversarial relationship, these systems become much easier to handle. In some cases, the adversarial relationship is explicit and direct, i.e., the upper-level DM's sole objective is to prevent the lower-level DM from achieving a known objective. Such systems, called *zero-sum*, arise in analyzing *interdiction problems* (see below).

Although bilevel linear programming (BLP) has received increased attention recently, the literature on IBLP remains scarce. Moore and Bard (1990) introduced a general framework for mixed integer bilevel linear programming (MIBLP),

described associated computational challenges, and suggested a branch-and-bound algorithm. The vast majority of the remaining IBLP literature has been restricted to various special cases. Bard and Moore (1992) developed a specialized algorithm for binary bilevel programs. Dempe (2001) considered the case characterized by continuous upper-level variables and integer lower-level variables and used a cutting plane approach to approximate the lower-level feasible region. Wen and Yang (1990) considered the opposite case, where the lower-level problem is a linear program and the upper-level problem is an integer program. Linear programming duality was used to derive exact and heuristic solutions.

A closely related class of models mentioned above that has already proven its utility in practice is that of the *interdiction models*. Most research on these models has focused on the *network interdiction problem* (Wollmer, 1964; McMasters and Mustin, 1970; Ghare et al, 1971; Wood, 1993; Cormican et al, 1998; Israeli and Wood, 2002; Held and Woodruff, 2005; Janjarassuk and Linderoth, 2006; Royset and Wood, 2007; Lim and Smith, 2007; Morton et al, 2007), in which the lower-level DM represents an entity operating a network of some sort. The upper-level DM (or interdictor) attempts to reduce the network performance as much as possible via the removal (complete or otherwise) of portions (subsets of arcs or nodes) of the network. Here, we generalize the underlying concept behind these network interdiction models by allowing the lower-level problem to be completely general and introducing the "interdiction" of lower-level decision variables in order to obtain a class of models with a much wider range of application.

The remainder of the paper is composed as follows. In Section 2, we describe the mathematical models that we consider. In Section 3, we discuss the challenge of solving these models and the barriers to generalizing solution methods for single-level mathematical programming problems. In Section 4, we describe how to overcome these challenges and propose a branch-and-cut algorithm for IBLPs. Section 5 illustrates the algorithm via an example and provides some preliminary computational results. Finally, in Section 6, we provide conclusions and directions for future work.

2 Definitions and Notation

When discussing various relaxations, it will sometimes be convenient to refer to the matrices $A := [(A^1)^\top | (A^2)^\top]^\top$ and $G := [0|(G^2)^\top]^\top$, and the vector $b := [(b^1)^\top | (b^2)^\top]^\top$. The region obtained by dropping the optimality requirement for the lower-level variables is then given by

$$\Omega^I = \{(x,y) \in \mathbb{Z}^{n_1} \times \mathbb{Z}^{n_2} \mid Ax + Gy \leq b, x, y \geq 0\}.$$

Removing the integrality requirements from Ω^I yields the polyhedral region

$$\Omega = \{(x,y) \in \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \mid Ax + Gy \leq b, x, y \geq 0\}.$$

For each upper-level solution $x \in \mathscr{P}_U \cap \mathbb{Z}^{n_1}$, we define the follower's *rational reaction set* to be

$$M^I(x) = \text{argmax}\{d^2 y \mid y \in \mathscr{P}_L(x) \cap \mathbb{Z}^{n_2}\}.$$

If we set

$$\Omega^I_{\text{proj}} = \left\{x \in \mathscr{P}_U \cap \mathbb{Z}^{n_1} \mid \exists y \text{ with } (x,y) \in \Omega^I\right\},$$

then any point (x,y) such that $x \in \Omega^I_{\text{proj}}$ and $y \in M^I(x)$ is called *bilevel feasible*. The IBLP problem can then be restated as that of determining

$$z_{IBLP} = \max_{(x,y)\in\mathscr{F}^I} c^1 x + d^1 y, \tag{IBLP}$$

where $\mathscr{F}^I = \left\{(x,y) \mid x \in \Omega^I_{\text{proj}}, y \in M^I(x)\right\}$. We also define the continuous analog of $\mathscr{F}^I$ by

$$\mathscr{F} = \left\{(x,y) \mid x \in \Omega_{\text{proj}}, y \in M(x)\right\},$$

where

$$\Omega_{\text{proj}} = \{x \in \mathscr{P}_U \mid \exists y \text{ with } (x,y) \in \Omega\}$$

and

$$M(x) = \text{argmax}\{d^2 y \mid y \in \mathscr{P}_L(x)\}.$$

Consistent with the existing literature (Bard, 1988; Bard and Moore, 1990; Moore and Bard, 1990; Bard and Moore, 1992), we assume that Ω^I is nonempty and bounded and that $\mathscr{P}_L(x) \cap \mathbb{Z}^{n_2} \neq \emptyset$ for all $x \in \mathscr{P}_U \cap \mathbb{Z}^{n_1}$. Further, we assume that if the lower-level DM's rational reaction set $M^I(x)$ is not a singleton, the upper-level DM is allowed to choose from among the alternatives one that is optimal with respect to the upper-level objective. This is the so-called *optimistic formulation* of the problem. The reader is referred to Loridan and Morgan (1996) for further insight on and discussion of alternative formulations.

3 Computational Challenges of IBLP

Because ILP is a special case of IBLP, it is clear that IBLP is also an $\mathscr{NP}$-hard problem. In fact, in contrast to the ILP case, the question of IBLP feasibility is not even in $\mathscr{NP}$, essentially because the question of whether a pair $(x,y) \in \mathbb{Z}^{n_1} \times \mathbb{Z}^{n_2}$ is feasible for a given IBLP is itself an ILP. Hansen et al (1992) show that even the continuous version of the problem (a BLP) is strongly $\mathscr{NP}$-hard and Vicente et al (1994) adds that checking local optimality for BLPs is an $\mathscr{NP}$-hard problem. All of this indicates that solving IBLPs in practice is likely to be extremely challenging.

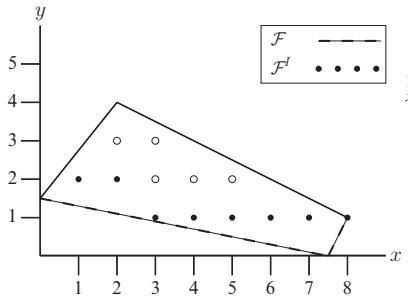
A natural approach to developing algorithms for solving IBLPs is to consider generalizations of the techniques that are used for ILPs. It does not take long, however, to realize that our intuition does not easily carry over from the case of ILP

to the case of IBLP. In a branch-and-bound algorithm for standard integer linear programming, integrality constraints are removed and the resulting linear program, which is easily seen to be a relaxation of the original ILP, is solved. The solution to this relaxed problem yields useful information about the original problem. In particular, we can make use of the following well-known rules to prune the branch and bound tree.

(R1) If the relaxed problem has no feasible solution, then neither does the original problem.
(R2) If the relaxed problem has a solution, then its value is a valid upper bound on the optimal value of the maximization original problem.
(R3) If the solution to the relaxed subproblem satisfies integrality restrictions, then it is optimal for the original problem.

Unfortunately, these rules cannot be extended in a straightforward way to IBLPs because dropping the integrality constraints from both upper and lower-level problems does not result in a relaxation, as the example in Figure 1 illustrates. In the figure, the polyhedron represents the set Ω, while the integer points in this polyhedron comprise the discrete set Ω^I. Within each of Ω and Ω^I, we have indicated points that satisfy the optimality constraint on the lower-level variables (i.e., the bilevel feasible solutions). From the figure, it is easy to see that $\mathscr{F} \subseteq \Omega$, $\mathscr{F}^I \subseteq \Omega^I$, and $\Omega^I \subseteq \Omega$. It is not the case, however, that $\mathscr{F}^I \subseteq \mathscr{F}$. Hence, the BLP obtained by dropping integrality constraints is not a relaxation of the IBLP.

In this example, optimizing over the continuous region $\mathscr{F}$ yields the integer solution $(8,1)$, with the upper-level objective value 18. However, the true solution to the IBLP is $(2,2)$, with objective value 22. From this, we observe that even when solutions to $\max_{(x,y)\in\mathscr{F}} c^1x+d^1y$ are in $\mathscr{F}^I$, they are not necessarily optimal. Thus, except in certain special cases, only Rule (R1) above remains valid if we simply remove integrality constraints from the IBLP to yield a BLP. Complicating matters further is the question of how to branch when faced with a solution that is integer but infeasible.



$$\begin{aligned} \max_{x\in\mathbb{Z}_+} \quad & x+10y \\ \text{s.t.} \quad & y \in \operatorname{argmax}\{-y : \\ & -25x+20y \le 30 \\ & x+2y \le 10 \\ & 2x-y \le 15 \\ & 2x+10y \ge 15 \\ & y \in \mathbb{Z}_+ \} \end{aligned}$$

Fig. 1 The feasible region of IBLP (Moore and Bard, 1990).

4 Branch and Cut

As with many classes of mathematical programs, the most obvious route to achieving global optimality is the development of bounding procedures that can be used to drive a branch-and-bound algorithm. As we have just observed, however, the bounding, fathoming, and branching procedures employed in traditional LP-based branch-and-bound algorithms cannot be applied in a straightforward way. In this section, we describe how to overcome these challenges to develop a generalized branch-and-cut algorithm for IBLPs that follows the same basic paradigm used in ILP. This work improves on the branch-and-bound algorithm originally suggested by Moore and Bard (1990) in a number of significant ways that we point out below.

4.1 Bounding

Although removing the integrality restrictions on all variables does not result in a valid relaxation, removing the lower-level optimality constraint from the problem *does* yield the relaxation

$$\max_{(x,y)\in\Omega^I} c^1x + d^1y, \tag{1}$$

similar to one suggested by Moore and Bard (1990). Unfortunately, as we noted earlier, determining whether solutions to this relaxation are bilevel feasible is a difficult problem in itself.

In order to improve upon the bounds yielded by (1) and to avoid the potential difficulties associated with being forced to branch when faced with an infeasible integer solution, we consider here a branch-and-cut algorithm based on the iterative generation of linear inequalities valid for $\mathscr{F}^I$ and augmentation of the linear system describing Ω until an optimal member of $\mathscr{F}^I$ is exposed or we choose to branch. The procedures we suggest are analogous to those used in the case of ILP but also address the fact that integer solutions may not be feasible in this setting.

4.2 Generating Valid Inequalities

An inequality defined by (π_1, π_2, π_0) is called a *valid inequality* for $\mathscr{F}^I$ if $\pi_1 x + \pi_2 y \leq \pi_0$ for all $(x,y) \in \mathscr{F}^I$. Unless $\text{conv}(\mathscr{F}^I) = \Omega$, there exist inequalities that are valid for $\mathscr{F}^I$, but are violated by some members of Ω. In order to generate these inequalities, we must use information not contained in the linear description of Ω. For a point $(x,y) \in \mathbb{Z}^{n_1} \times \mathbb{Z}^{n_2}$ to be feasible for an IBLP, it must satisfy three conditions:

(C1) $(x,y) \in \Omega$,
(C2) $(x,y) \in \mathbb{Z}^{n_1} \times \mathbb{Z}^{n_2}$, and
(C3) $y \in M^I(x)$.

This is in contrast to standard ILPs, where we have only the first two conditions.

Because the first requirement is enforced by requiring membership in Ω, we must derive valid inequalities from the other two conditions. We start with the following straightforward, but useful observations.

Observation 1 *If the inequality (π_1, π_2, π_0) is valid for Ω^I, it is also valid for $\mathscr{F}^I$.*

Observation 2 *Let $(x,y) \in \Omega$ such that $y \notin M^I(x)$. If the inequality (π_1, π_2, π_0) is valid for $\Omega^I \setminus \{(x,y)\}$, it is also valid for $\mathscr{F}^I$.*

Observation 1 is derived from the relationship $\mathscr{F}^I \subseteq \Omega^I$ and allows us to separate fractional solutions to the LP resulting from removal of the lower-level optimality and integrality restrictions. Observation 2 states that we can separate points that are integer but not bilevel feasible. From these observations, we can derive two classes of valid inequalities to be used in a cutting plane procedure.

To initialize the cutting plane procedure, we must first solve the relaxation

$$\max_{(x,y)\in\Omega} c^1 x + d^1 y. \tag{LR}$$

If the solution $(\hat{x}, \hat{y})$ to (LR) does not satisfy condition (C2) above, we may apply standard cutting plane procedures used to separate points in $\Omega \setminus \Omega^I$ from $\Omega^I \supseteq \mathscr{F}^I$. For an overview of the various classes of valid inequalities used for separating fractional solutions from the convex hull of solutions to generic integer programs, see Cornuejols (2008). Any of the existing classes of valid inequalities are potential candidates for employment here, though the structure of each specific instance could be used to decide which classes are likely to be the most effective.

If $(\hat{x}, \hat{y})$ satisfies condition (C2), then we must check whether it satisfies condition (C3). This is done by solving the lower-level problem

$$\max_{y\in\mathscr{P}_L(x)\cap\mathbb{Z}^{n_2}} d^2 y \tag{2}$$

with the fixed upper-level solution $\hat{x}$. Let the solution to this IP be y^*. If $d^2\hat{y} = d^2 y^*$, then $\hat{y}$ is also optimal for (2) and we conclude that $(\hat{x}, \hat{y})$ is bilevel feasible. Otherwise, we must again generate an inequality separating $(\hat{x}, \hat{y})$ from $\mathscr{F}^I$. In either case, however, $(\hat{x}, y^*)$ is bilevel feasible and provides a valid lower bound on the optimal solution value of the original IBLP.

Now suppose $d^2\hat{y} < d^2 y^*$. In this case, $(\hat{x}, \hat{y})$ does not satisfy condition (C3) and is therefore not bilevel feasible. We may still use $(\hat{x}, y^*)$ to bound the original problem, but we would like to add an inequality to (LR) that is valid for $\mathscr{F}^I$ and violated by $(\hat{x}, \hat{y})$. The simple procedure encapsulated in the following proposition can be used to generate such an inequality.

Proposition 1. *Let $(\hat{x}, \hat{y}) \in \Omega^I$ be a basic feasible solution to* (LR)*. Let J be the set of constraints that are binding at $(\hat{x}, \hat{y})$. Then*

$$\pi_1 x + \pi_2 y \leq \pi_0 - 1, \tag{3}$$

where $(\pi_1, \pi_2) = \sum_{j\in J}(a_j, g_j)$ and $\pi_0 = \sum_{j\in J} b_j$, is valid for $\mathscr{F}^I$.

Proof. The fact that $(\hat{x},\hat{y})$ is a basic feasible implies that there exist $n = n_1 + n_2$ linearly independent constraints in the description of Ω that are binding at $(\hat{x},\hat{y})$. Thus, the system $a'_j x + g'_j y = b_j, j \in J$ has a unique solution, namely $(\hat{x},\hat{y})$. This, in turn, implies that $(\hat{x},\hat{y})$ is the unique point of intersection between the hyperplane defined by the equation $\pi_1 x + \pi_2 y = \pi_0$ and the set Ω^I. It follows that the inequality $\pi_1 x + \pi_2 y \leq \pi_0$ is valid for Ω^I. Because the face of Ω induced by this inequality does not contain any other members of Ω^I and there does not exist $(x,y) \in \mathbb{Z}^{n_1} \times \mathbb{Z}^{n_2}$ such that $\pi_1 x + \pi_2 y \in (\pi_0 - 1, \pi_0)$, this implies that the inequality $\pi_1 x + \pi_2 y \leq \pi_0 - 1$ is valid for $\Omega^I \setminus \{(\hat{x},\hat{y})\}$. Applying Observation 2 yields the result. □

An example is shown in Figure 2 for the instance

$$\max_x \min_y \{y \mid -x + y \leq 2, -2x - y \leq -2, 3x - y \leq 3, y \leq 3, x, y \in \mathbb{Z}_+\}.$$

In the figure, we can see the bilevel feasible region $\mathcal{F}^I = \{(0,2),(1,0),(2,3)\}$. Also shown in the figure is the bilevel feasible region $\mathcal{F}$ of the corresponding BLP. In this example, we start with the integer point $(1,3)$, an optimal solution to the LP

$$\max_{x,y} \{y \mid -x + y \leq 2, -2x - y \leq -2, 3x - y \leq 3, y \leq 3, x, y \in \mathbb{R}_+\}.$$

It is easy to see that this point is not bilevel feasible, because the rational choice for the lower-level DM would be $y = 0$, when $x = 1$. Thus, we require a cut that separates $(1,3)$. Combining the constraints active at $(1,3)$ yields the half-space $\{(x,y) \in \mathbb{Z}^{n_1} \times \mathbb{Z}^{n_2} \mid -x + 2y \leq 5\}$ and applying the procedure described above, we obtain the new inequality

$$-x + 2y \leq 4,$$

which is valid for $\mathcal{F}^I$, but not satisfied by $(1,3)$. Note that after adding this cut, the optimal solution is obtained in the next iteration. Without the cutting plane procedure we have just described, we would be forced to branch after producing this solution in a branch-and-bound framework.

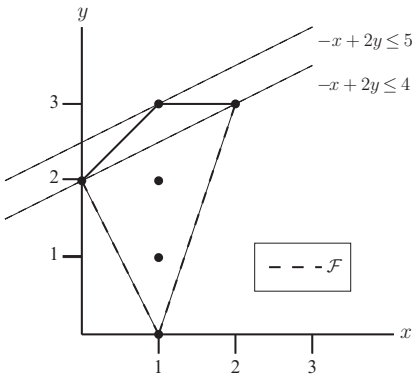


Fig. 2 An example of the bilevel feasibility cut.

The combination of this procedure, the bounding technique of Section 4.1, and the branching techniques given in Section 4.3 yields a branch-and-cut algorithm. However, it is clear that the procedure will fail on large-scale problems. In order to solve problems of interesting size, additional classes of valid inequalities derived from Condition (C3) are necessary. One such class that utilizes information from the value function of the lower-level ILP is described in DeNegre et al (2008).

4.3 Branching

As we have just described, an important advantage of our algorithm over its predecessor from Moore and Bard (1990), is the fact that we are not forced to branch after producing an infeasible integer solution and are therefore free to employ the well-developed branching strategies used in algorithms for traditional ILP, such as strong branching, pseudocost branching, or the recently introduced reliability branching (Achterberg et al, 2005). Of course, it is also possible to branch using disjunctions obtained from violations of Condition (C3). Although this is unnecessary for small problems, we believe such branching strategies may ultimately be necessary for larger problems. Specialized branching techniques for bilevel problems are discussed in DeNegre et al (2008).

4.4 A Branch-and-cut Algorithm

Putting together the procedures of the preceding three sections, we obtain a branch-and-cut algorithm that consists of solving the linear relaxation (LR), iteratively generating valid inequalities to improve the bound, and branching when necessary. In addition to the obvious advantage of producing potentially improved bounds, an advantage of this approach over the one proposed by Moore and Bard (1990) is that it relies only on the solution of standard ILPs and preserves all the usual rules of fathoming and branching. It therefore allows us to immediately leverage our vast knowledge of how to solve standard ILPs. The general framework of such an algorithm is described next.

Let

$$\max_{(x,y)\in\mathscr{F}_t} c^1x + d^1y. \tag{IBLP^t}$$

be the IBLP defined at node t of the branch-and-cut tree. To process node t, we first solve the LP

$$z^t_{\mathrm{LP}} = \max_{(x,y)\in\Omega_t} c^1x + d^1y. \tag{LP^t}$$

and denote its solution by (x^t, y^t) (if it exists). If either the LP is infeasible or the optimal value of (LP^t) is less than the current lower bound L, we can fathom the current node. Otherwise we generate valid inequalities to separate the current solution

from $\mathscr{F}^I$. If $(x^t, y^t) \in \Omega^I$, we check for bilevel feasibility. If the solution is feasible, we can stop. Otherwise, we add cuts of the form (3) to separate the current solution from $\Omega^I \setminus \{(x^t, y^t)\}$ if necessary and iterate. If a fractional solution is found, we either add cuts to separate the current solution from $\Omega^t \cap \mathbb{Z}^{n_1} \times \mathbb{Z}^{n_2}$ and iterate or else we branch. A general outline of the node processing subroutine is given in Algorithm 1.

Algorithm 1 Node Processing Loop

1: Solve (LP^t). If (LP^t) has an optimal solution, denote it (x^t, y^t). Then, depending on the outcome, do the following:

- If (LP^t) is infeasible, so is (IBLP^t) and the current node can be pruned.
- Else, if $z^t_{\text{LP}} \leq L$, the current node can be pruned.
- Else, go to Step 2.

2: Generate valid inequalities.

- If $(x^t, y^t) \in \Omega^I$, fix $x \leftarrow x^t$, and solve $z^t_{\text{LL}} = \max_{y \in \mathscr{P}_L(x^t) \cap \mathbb{Z}^{n_2}} d^2 y$. If $z^t_{\text{LL}} = d^2 y^t$ set $L \leftarrow c^1 x^t + d^1 y^t$ and prune the current node; else set

$$\Omega_{t+1} = \Omega_t \cap \left\{ (x,y) \in \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \mid \sum_{j \in J} (a_j x + g_j y) \leq \sum_{j \in J} b_j - 1 \right\},$$

  where J is the set of active constraints in Ω_t at (x^t, y^t), set $t \leftarrow t+1$, and go to Step 1.
- Else, either generate and add cuts valid for $\Omega^t \cap \mathbb{Z}^{n_1} \times \mathbb{Z}^{n_2}$ and go to Step 1 or BRANCH.

4.5 Specialized Methods for Binary IBLPs

The bilevel feasibility cut (3) ensures that bilevel infeasible solutions to (1) are not generated in future iterations. However, by design, it does not cut off any other integer points. This may result in the generation of a sequence of points $(x^*, y^1), (x^*, y^2), \ldots, (x^*, y^k)$ such that $y^i \notin M^I(x^*)$ for $i < k$. If $x \in \mathbb{B}^{n_1}$, information obtained from the lower-level problem can be used to avoid this problem. While checking bilevel feasibility, we obtain an optimal solution and associated objective value $z_L(x^*)$ for the lower-level problem (2). This leads to the implication $x = x^* \Rightarrow d^2 y \geq z_L(x^*)$. Let $I_0 := \{i \mid x^*_i = 0\}$ and $I_1 := \{i \mid x^*_i = 1\}$. Note that for $x \in \mathbb{B}^{n_1}$, we have that $\sum_{i \in I_0} x_i + \sum_{i \in I_1} (1 - x_i) = 0$ if and only if $x = x^*$. Otherwise, $\sum_{i \in I_0} x_i + \sum_{i \in I_1} (1 - x_i) \geq 1$. One way to model this implication is to introduce the constraint $\sum_{i \in I_0} x_i + \sum_{i \in I_1} (1 - x_i) + \delta \geq 1$, where $\delta \in \mathbb{B}$, which imposes the implication $x = x^* \Rightarrow \delta = 1$. Then, adding the constraint $d^2 y + m\delta \geq m + z_L(x^*)$, where $m = \min\{d^2 y - z_L(x^*) \mid (x,y) \in \Omega^I\}$, enforces $\delta = 1 \Rightarrow d^2 y \geq z_L(x^*)$, as desired. Exploring further such logical implications is an area of future research.

5 Computational Results

The branch-and-cut algorithm was implemented in C++, utilizing standard software components available from the Computational Infrastructure for Operations Research (COIN-OR) repository (Lougee-Heimer, 2003). The implementation uses the COIN-OR High Performance Parallel Search Framework (CHiPPS) to perform branch and bound, the MILP solver framework BLIS (part of CHiPPS), the COIN-OR LP Solver (CLP) for solving the LPs that arise in branch and cut, the SYMPHONY MILP solver for solving the lower-level ILPs, the Cut Generation Library (CGL) for generating cutting planes, and the Open Solver Interface (OSI) for interfacing with CHiPPS and SYMPHONY.

To our knowledge, the only other general IBLP algorithm proposed in the literature has been that of Moore and Bard (1990). We do not have the test set of Moore and Bard (1990) or an implementation of their algorithm available, so a comprehensive comparison to their algorithm is not feasible. In order to provide *some* basis for comparison, we did examine the branch-and-cut tree constructed by our algorithm on one of the examples from their original paper. The feasible region of the IBLP and our branch-and-cut tree are shown in Figure 3. In this simple case, our algorithm generated a total of seven nodes, and processed five, while the same example in the original paper required twelve nodes. Of course, this comparison is only a single instance, but examination of the two search trees does provide some evidence for our intuition that certain aspects of Moore and Bard's algorithm, such as the requirement to branch on integer variables, result in a less efficient search.

We also tested our algorithm on a set of interdiction problems, in which the lower-level problems were binary knapsack problems with a single constraint. The goal of the upper-level DM was to minimize the maximum profit achievable by the lower-level DM by fixing a subset of the variables in the lower-level problem to zero. A cost was associated with the fixing of each lower-level variable to zero and the upper-level problems contained a single constraint, representing the available interdiction budget. To create these instances, data files describing bicriteria

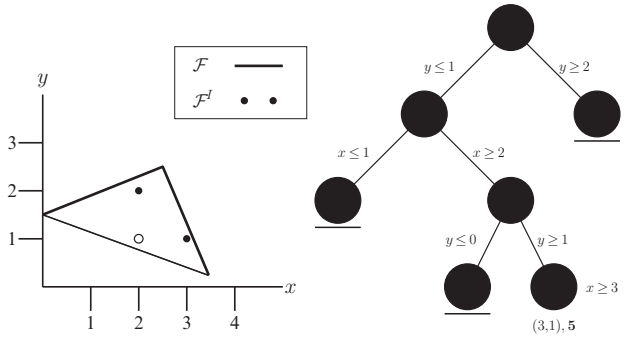


Fig. 3 Example 2 from Moore and Bard (1990) and the resulting branch-and-cut tree.

Table 1 Summary results from the knapsack interdiction.

	Maximum Infeasibility			Strong Branching		
$2n$	Avg Nodes	Avg Depth	Avg CPU (s)	Avg Nodes	Avg Depth	Avg CPU (s)
20	1801	16.45	3.17	1125	16.95	4.69
22	3538	18.25	6.63	1860	17.40	9.13
24	7034	20.20	13.27	3314	19.65	17.50
26	13867	22.00	27.54	6294	20.20	35.84
28	26155	23.85	60.08	11915	23.00	71.90
30	60626	26.65	124.84	23917	24.15	145.99
32	125840	26.75	249.19	45879	25.80	296.16
34	253965	29.65	516.65	–	–	–

knapsack problems were taken from the *Multiple Criteria Decision Making* library (Figueira, 2000). The first objective in each file was used to define a lower-level objective function, while the second objective provided a budget constraint. The available budget was then chosen to be $\lceil \sum_{i=1}^{n} a_i/2 \rceil$, where a_i is the cost of interdicting lower-level variable i. For a knapsack problem with n items, this construction yielded a problem with $2n$ variables and $n+2$ constraints. All computational tests were performed on an Intel Xeon 2.4GHz processor with 4GB of memory. Summarized results of two sets of runs—one in which we used maximum infeasibility branching to select branching candidates and one in which we used strong branching, are shown in Table 1, where the results for each problem size reflect the average of 20 instances. Note that a dash indicates that no instances of the corresponding size were able to be solved due to memory requirements. These results look promising, but are preliminary at best. For these instances, strong branching reduced the size of the search tree significantly, but required more computation time. More fine-tuning of algorithm parameters is needed to determine the best branching strategy.

6 Conclusions

We have discussed the challenges associated with solving integer bilevel linear programming problems and described a branch-and-cut algorithm that can be seen as a generalization of the familiar algorithm used for solution of standard integer linear programs. The primary advantage of this approach is the ability to exploit the vast array of existing technology for solving ILPs techniques. The next step in the development of this approach is to include a wider range of the supplemental techniques that have proven critical in our ability to solve difficult integer linear programs in practice. These include such improvements as the development of preprocessing techniques, primal heuristics, additional classes of valid inequalities, branching rules based on disjunctions involving more than one variable, and more effective search strategies. In this paper, we have only suggested a starting point and much work remains to be done to make these methods practical for large-scale instances.

References

Achterberg T, Koch T, Martin A (2005) Branching rules revisited. Operations Research Letters 33(1):42–54

Bard J (1988) Convex two-level optimization. Mathematical Programming 40: 15–27

Bard J, Moore J (1990) A branch and bound algorithm for the bilevel programming problem. SIAM Journal on Scientific and Statistical Computing 11(2):281–292

Bard J, Moore J (1992) An algorithm for the discrete bilevel programming problem. Naval Research Logistics 39:419–435

Cormican K, Morton D, Wood R (1998) Stochastic network interdiction. Operations Research 46(2):184–197

Cornuejols G (2008) Valid inequalities for mixed integer linear programs. Mathematical Programming B 112:3–44

Dempe S (2001) Discrete bilevel optimization problems. Tech. Rep. D-04109, Institut fur Wirtschaftsinformatik, Universitat Leipzig, Leipzig, Germany

DeNegre S, Ralphs T, Guzelsoy M (2008) A new class of valid inequalities for mixed integer bilevel linear programs. Tech. rep., Lehigh University

Figueira J (2000) MCDM Numerical Instances Library. URL http://www.univ-valenciennes.fr/ROAD/MCDM/ListMOKP.html

Ghare P, Montgomery D, Turner W (1971) Optimal interdiction policy for a flow network. Naval Research Logistics Quarterly 18:27–45

Hansen P, Jaumard B, Savard G (1992) New branch-and-bound rules for linear bilevel programming. SIAM Journal on Scientific and Statistical Computing 13(5):1194–1217

Held H, Woodruff D (2005) Heuristics for multi-stage interdiction of stochastic networks. Journal of Heuristics 11(5-6):483–500

Israeli E, Wood R (2002) Shortest path network interdiction. Networks 40(2): 97–111

Janjarassuk U, Linderoth J (2006) Reformulation and sampling to solve a stochastic network interdiction problem. Tech. Rep. 06T-001, Lehigh University

Lim C, Smith J (2007) Algorithms for discrete and continuous multicommodity flow network interdiction problems. IIE Transactions 39(1):15–26

Loridan P, Morgan J (1996) Weak via strong stackelberg problem: New results. Journal of Global Optimization 8(3):263–287

Lougee-Heimer R (2003) The Common OPtimization INterface for Operations Research. IBM Journal of Research and Development 47(1):57–66

McMasters A, Mustin T (1970) Optimal interdiction of a supply network. Naval Research Logistics Quarterly 17:261–268

Moore J, Bard J (1990) The mixed integer linear bilevel programming problem. Operations Research 38(5):911–921

Morton D, Pan F, Saeger K (2007) Models for nuclear smuggling interdiction. IIE Transactions 39(1):3–14

Royset J, Wood R (2007) Solving the bi-objective maximum-flow network-interdiction problem. INFORMS Journal on Computing 19(2):175–184

Vicente L, Savard G, Judice J (1994) Descent approaches for quadratic bilevel programming. Journal of Optimization Theory and Applications 81:379–399

Wen U, Yang Y (1990) Algorithms for solving the mixed integer two-level linear programming problem. Computers & Operations Research 17(2):133–142

Wollmer R (1964) Removing arcs from a network. Operations Research 12(6): 934–940

Wood R (1993) Deterministic network interdiction. Mathematical and Computer Modelling 17(2):1–18