Preprocessing and Probing Techniques for Mixed Integer Programming Problems

M.W.P. Savelsbergh
School of Industrial and Systems Engineering
Georgia Institute of Technology
Atlanta, GA 30332-0205

Email: Martin.Savelsbergh@isye.gatech.edu

Abstract

In the first part of the paper, we present a framework for describing basic techniques to improve the representation of a mixed integer programming problem. We elaborate on identification of infeasibility and redundancy, improvement of bounds and coefficients, and fixing of binary variables. In the second part of the paper, we discuss recent extensions to these basic techniques and elaborate on the investigation and possible uses of logical consequences.

Subject Classification: Programming, Integer
Other key words: Preprocessing, Probing

The success of branch-and-cut algorithms for combinatorial optimization problems [5, 10] and large scale 0-1 linear programming problems [2] has lead to a renewed interest in mixed integer programming. The key idea of the branch-and-cut approach is reformulation. Problems are reformulated so as to make the difference in the objective function values between the solutions to the linear programming relaxation and the integer program as small as possible.

There are various ways to tighten the linear programming relaxation of an integer program. Preprocessing and probing techniques [1, 3, 4, 6, 7] try, among others things, to reduce the size of coefficients in the constraint matrix and to reduce the size of bounds on the variables. Constraint generation techniques [2, 11] try to generate strong valid inequalities. Johnson [9] discusses a wide range of issues related to modeling and strong linear programs for mixed integer programming.

It is well known, that there are many ways to represent a mixed integer program by linear inequalities while guaranteeing that the underlying set of feasible solutions is unchanged. In the first part of this paper, we present a framework for describing various techniques that modify a given representation of a mixed integer programming problem in such a way that the set of feasible solutions of the linear programming relaxation is reduced, but the set of feasible solutions to the mixed integer program is not affected. This may reduce the integrality gap, i.e., the difference between the objective function values of the linear programming relaxation and the integer program, which is crucial in the context of a linear programming based branch and bound algorithm. We concentrate on identifying infeasibility and redundancy, improving bounds and coefficients, and fixing variables.

Several other papers have been written on this subject, most notably Dietrich and Escudero [3] and Hoffman and Padberg [6]. Dietrich and Escudero consider coefficient reduction for 0-1 linear programming problems containing variable upper bound constraints and Hoffman and Padberg discuss the implementation of coefficient reduction for 0-1 linear programming problems containing special ordered set constraints. Both papers deal with pure 0-1 linear programming problems and in both papers the general ideas are somewhat obscured by the specific perspective.

The purpose of this paper is twofold. First, to introduce a framework for describing preprocessing and probing techniques for mixed integer programming problems and survey some of the well-known basic techniques. In doing so, we clearly separate the underlying ideas from the implementation issues. Second, to present some of the, more recently developed, techniques that are currently employed by the state-of-the-art general purpose mixed integer optimizers.

1 Basic preprocessing and probing techniques

The underlying idea of the basic techniques to improve a given representation of a mixed integer programming problem $\min\{cx + hy : Ax + Gy \leq b, x \in \{0,1\}^n, y \in RR^m\}$ is to analyze each of the inequalities of the system of inequalities defining the feasible region in turn, trying to establish whether the inequality forces the feasible region to be empty, whether the inequality is redundant, whether the inequality can be used to improve the bounds on the variables, whether the inequality can be strengthened by modifying its coefficients, or whether the inequality forces some of the binary variables to either zero or one.

We assume that the inequality currently under consideration is of the form

$$\sum_{j \in B^+} a_j x_j - \sum_{j \in B^-} a_j x_j + \sum_{j \in C^+} g_j y_j - \sum_{j \in C^-} g_j y_j \leq b,$$

where $B = B^+ \cup B^-$ is the set of binary variables, $C = C^+ \cup C^-$ is the set of both integer and continuous variables, and $a_j > 0$ for $j \in B$ and $g_j > 0$ for $j \in C$. Note that we do not distinguish integer and continuous variables. Furthermore, we assume that the lower and upper bounds of integer and continuous variables are denoted by l_j and u_j.

For the remainder of this section, let $Ax + Gy \leq b, x \in \{0,1\}^n, y \in RR^m$ be a given representation of a mixed integer programming problem, let $a^i x + g^i y \leq b_i$ be any inequality of the system, and let $A^i x + G^i y \leq b^i$ denote the system of inequalities obtained from $Ax + Gy \leq b$ by deleting row $a^i x + g^i y \leq b_i$.

1.1 Basic preprocessing techniques

Identification of infeasibility
Consider the following mixed integer programming problem

$$z = \min \sum_{j \in B+} a^i_j x_j - \sum_{j \in B-} a^i_j x_j + \sum_{j \in C+} g^i_j y_j - \sum_{j \in C-} g^i_j y_j$$

subject to

$$A^i x + G^i y \leq b^i$$

$$l \leq y \leq u$$

$$x \in \{0,1\}^n, y \in RR^m.$$

If $z > b_i$, then obviously the feasible region is empty. Unfortunately, the above mixed integer programming problem is as hard to solve as the original problem, but it should be clear that any lower bound on z suffices ($z \geq z_{LB} > b_i$). The simplest, but also weakest, lower bound is obtained by completely discarding the system $A^i x + G^i y \leq b^i$. In that case, we can conclude that the problem is infeasible if

$$-\sum_{j \in B-} a^i_j + \sum_{j \in C+} g^i_j l_j - \sum_{j \in C-} g^i_j u_j > b_i.$$

Identification of redundancy
Consider the following mixed integer programming problem

$$z = \max \sum_{j \in B+} a^i_j x_j - \sum_{j \in B-} a^i_j x_j + \sum_{j \in C+} g^i_j y_j - \sum_{j \in C-} g^i_j y_j$$

subject to

$$A^i x + G^i y \leq b^i$$

$$l \leq y \leq u$$

$$x \in \{0,1\}^n, y \in RR^m.$$

If $z \leq b_i$, then obviously the inequality $a^i x + g^i y \leq b_i$ is redundant. Of course, the above mixed integer programming problem is as hard to solve as the original problem, but it should be clear that any upper bound on z suffices ($z \leq z_{UB} \leq b_i$). The simplest, but also weakest, upper bound is obtained by completely discarding the system $A^i x + G^i y \leq b^i$. In that case, we can conclude that the inequality is redundant if

$$\sum_{j \in B^+} a^i_j + \sum_{j \in C^+} g^i_j u_j - \sum_{j \in C^-} g^i_j l_j \leq b_i.$$

Improving bounds

Consider a variable y_k, $k \in C^+$ and the following integer programming problem

$$z_k = \min \sum_{j \in B^+} a^i_j x_j - \sum_{j \in B^-} a^i_j x_j + \sum_{j \in C^+ \setminus \{k\}} g^i_j y_j - \sum_{j \in C^-} g^i_j y_j$$

subject to

$$A^i x + G^i y \leq b^i$$

$$l \leq y \leq u$$

$$x \in \{0,1\}^n, y \in RR^m.$$

Clearly, $y_k \leq (b_i - z_k)/g^i_k$. Therefore, the upper bound u_k on variable y_k, $k \in C^+$ can be improved if $(b_i - z_k)/g^i_k < u_k$. When we discard the system $A^i x + G^i y \leq b^i$, we can conclude that the upper bound on variable y_k, $k \in C^+$ can be improved if

$$(b_i + \sum_{j \in B^-} a^i_j - \sum_{j \in C^+ \setminus \{k\}} g^i_j l_j + \sum_{j \in C^-} g^i_j u_j)/g^i_k < u_k$$

Next, consider a variable y_k, $k \in C^-$ and the following integer programming problem

$$z_k = \min \sum_{j \in B^+} a^i_j x_j - \sum_{j \in B^-} a^i_j x_j + \sum_{j \in C^+} g^i_j y_j - \sum_{j \in C^- \setminus \{k\}} g^i_j y_j$$

subject to

$$A^i x + G^i y \leq b^i$$

$$l \leq y \leq u$$

$$x \in \{0,1\}^n, y \in RR^m.$$

Obviously, $y_k \geq (z_k - b_i)/g^i_k$. Therefore, the lower bound l_k on a variable y_k, $k \in C^-$ can be improved if $(z_k - b_i)/g^i_k > l_k$. When we discard the system $A^i x + G^i y \leq b^i$, we can conclude that the lower bound on variable y_k, $k \in C^-$ can be improved if

$$(-\sum_{j \in B^-} a^i_j + \sum_{j \in C^+} g^i_j l_j - \sum_{j \in C^- \setminus \{k\}} g^i_j u_j - b_i)/g^i_k > l_k$$

1.2 Basic probing techniques

Probing techniques are based on the investigation of logical consequences, i.e., tentatively setting a binary variable x_k to either 0 or 1 and exploring the consequences. In the framework presented above, this amounts to adding the constraint $x_k = 0$ or $x_k = 1$ to the set of constraints and applying the basic preprocessing techniques to this extended formulation. Obviously, the findings have to be interpreted differently.

Fixing variables
Consider a binary variable $x_k, k \in B^+$ and the following extended mixed integer programming problem

$$z_k = \min \sum_{j \in B^+} a^i_j x_j - \sum_{j \in B^-} a^i_j x_j + \sum_{j \in C^+} g^i_j y_j - \sum_{j \in C^-} g^i_j y_j$$

subject to

$$A^i x + G^i y \leq b^i$$

$$x_k = 1$$

$$l \leq y \leq u$$

$$x \in \{0,1\}^n, y \in RR^m.$$

In case $z_k > b_i$, then obviously the feasible region of this extended formulation is empty. Therefore, $x_k \neq 1$ in any feasible solution of the original formulation and x_k can be fixed to 0. When we discard the system $A^i x + G^i y \leq b^i$, we can fix the variable x_k to 0 if

$$a^i_k - \sum_{j \in B^-} a^i_j + \sum_{j \in C^+} g^i_j l_j - \sum_{j \in C^-} g^i_j u_j > b_i.$$

Next, consider a binary variable $x_k, k \in B^-$ and the following extended mixed integer programming problem

$$z_k = \min \sum_{j \in B^+} a^i_j x_j - \sum_{j \in B^-} a^i_j x_j + \sum_{j \in C^+} g^i_j y_j - \sum_{j \in C^-} g^i_j y_j$$

subject to

$$A^i x + G^i y \leq b^i$$

$$x_k = 0$$

$$l \leq y \leq u$$

$$x \in \{0,1\}^n, y \in RR^m.$$

If $z_k > b_i$, then obviously the feasible region of this extended formulation is empty. Therefore, $x_k \neq 0$ in any feasible solution of the original formulation and x_k can be fixed to 1. When we discard the system $A^i x + G^i y \leq b^i$, we can fix the variable x_k to 1 if

$$-\sum_{j \in B^- \setminus \{k\}} a_j^i + \sum_{j \in C^+} g_j^i l_j - \sum_{j \in C^-} g_j^i u_j > b_i.$$

Improving coefficients
Consider a binary variable $x_k, k \in B^+$ and the following extended mixed integer programming problem

$$z_k = \max \sum_{j \in B^+} a_j^i x_j - \sum_{j \in B^-} a_j^i x_j + \sum_{j \in C^+} g_j^i y_j - \sum_{j \in C^-} g_j^i y_j$$

subject to

$$A^i x + G^i y \leq b^i$$

$$x_k = 0$$

$$l \leq y \leq u$$

$$x \in \{0,1\}^n, y \in RR^m.$$

If $z_k \leq b_i$, then obviously the inequality $a^i x + g^i y \leq b_i$ is redundant in this extended formulation. Consequently, under the assumption that $x_k = 0$, the set of feasible solutions is not affected if b_i and a_k^i are reduced by $\delta = b_i - z_k$. Furthermore, also under the assumption that $x_k = 1$, the set of feasible solutions is not affected if b_i and a_k^i are reduced by $\delta = b_i - z_k$. To see this, rewrite the inequality $a^i x + g^i y \leq b_i$ as

$$\sum_{j \in B^+ \setminus \{k\}} a_j^i x_j - \sum_{j \in B^-} a_j^i x_j + \sum_{j \in C^+} g_j^i y_j - \sum_{j \in C^-} g_j^i y_j \leq b_i - a_k^i \Rightarrow$$

$$\sum_{j \in B^+ \setminus \{k\}} a_j^i x_j - \sum_{j \in B^-} a_j^i x_j + \sum_{j \in C^+} g_j^i y_j - \sum_{j \in C^-} g_j^i y_j \leq (b_i - \delta) - (a_k^i - \delta) \qquad \delta \in RR_+ \Rightarrow$$

$$\sum_{j \in B^+ \setminus \{k\}} a_j^i x_j + (a_k^i - \delta) x_k - \sum_{j \in B^-} a_j^i x_j + \sum_{j \in C^+} g_j^i y_j - \sum_{j \in C^-} g_j^i y_j \leq b_i - \delta \qquad \delta \in RR_+.$$

Therefore, a_k and b_i can be decreased by $b_i - z_k$ without changing the set of feasible solutions. When we discard the system $A^i x + G^i y \leq b^i$, we can decrease a_k and b_i if

$$\sum_{j \in B^+ \setminus \{k\}} a^i_j + \sum_{j \in C^+} g^i_j u_j - \sum_{j \in C^-} g^i_j l_j < b_i.$$

Next, consider a binary variable $x_k, k \in B^-$ and the following extended mixed integer programming problem

$$z_k = \max \sum_{j \in B^+} a^i_j x_j - \sum_{j \in B^-} a^i_j x_j + \sum_{j \in C^+} g^i_j y_j - \sum_{j \in C^-} g^i_j y_j$$

subject to

$$A^i x + G^i y \leq b^i$$

$$x_k = 1$$

$$l \leq y \leq u$$

$$x \in \{0,1\}^n, y \in RR^m.$$

If $z_k \leq b_i$, then obviously the inequality $a^i x + g^i y \leq b_i$ is redundant in this extended formulation. Consequently, under the assumption that $x_k = 1$, the set of feasible solutions is not affected if a^i_k is decreased by $\delta = b_i - z_k$. Furthermore, also under the assumption that $x_k = 0$, the set of feasible solutions is not affected if a^i_k is decreased by $\delta = b_i - z_k$.

Therefore, a_k can be decreased by $b_i - z_k$ without changing the set of feasible solutions. When we discard the system $A^i x + G^i y \leq b^i$, we can decrease a_k if

$$\sum_{j \in B^+} a^i_j - a^i_k + \sum_{j \in C+} g^i_j u_j - \sum_{j \in C-} g^i_j l_j < b_i.$$

Observe that if the simplest bounds are used, i.e., the system $A^i x + G^i y \leq b^i$ is discarded, only variables appearing in the constraint under consideration can be fixed and only coefficients of variables appearing in the constraint under consideration can be modified.

1.3 Implementation

The basic preprocessing and probing techniques can be implemented very efficiently. Define

$$L^i_{\max} = \sum_{j \in B^+} a^i_j + \sum_{j \in C^+} g^i_j u_j - \sum_{j \in C^-} g^i_j l_j$$

and

$$L^i_{\min} = -\sum_{j \in B^-} a^i_j + \sum_{j \in C^+} g^i_j l_j - \sum_{j \in C^-} g^i_j u_j,$$

i.e., the maximum and minimum value of the summation on left-hand-side of the inequality under consideration. It is not hard to see that the basic preprocessing and probing techniques require the following tests

- identification of infeasibility:

$$L^i_{\min} > b_i$$

- identification of redundancy:

$$L^i_{\max} \leq b_i$$

- improvement of bounds:

$$(b_i - (L^i_{\min} - g^i_k l_k))/g^i_k < u_k \qquad y_k,\ k \in C^+$$

$$((L^i_{\min} + g^i_k u_k) - b_i)/g^i_k > l_k \qquad y_k,\ k \in C^-$$

- fixing of variables:

$$L^i_{\min} + a^i_k > b_i \qquad x_k,\ k \in B^+ \cup B^-$$

- improvement of coefficients:

$$L^i_{\max} - a^i_k < b_i \qquad x_k,\ k \in B^+ \cup B^-$$

Obviously, all these tests can be performed in constant time once the values $L^i_{\max}$ and $L^i_{\min}$ have been computed. As a consequence, the application of the basic preprocessing and probing techniques to all the inequalities of the system $Ax + Gy \leq b$ requires $O(n)$ time, where n is the total number of nonzero's in the system.

This concludes our discussion of the basic techniques to improve a given representation of a mixed integer programming problem. We end this section with some observations. We have assumed, without explicitly stating it, that we are always examining an inequality. The techniques have to be modified slightly in case of an equality. The techniques have been presented as simple and efficiently solvable approximations of mixed integer programming problems. The simplicity and computational efficiency is a consequence of restricting attention to information in a single inequality, i.e., discarding the information contained in the system $A^i x + G^i y \leq b^i$.

2 Advanced preprocessing and probing techniques

2.1 Special substructures

As observed in the previous section, the computational efficiency is a consequence of restricting attention to information in a single constraint. However, there is a trade off between efficiency and effectivity; more effective, but computationally less efficient, techniques are obtained when information from more than one inequality is used.

Dietrich and Escudero [3] and Hoffman and Padberg [6] investigate the trade off between efficiency and effectivity for special substructures, consisting of more than one constraint, that often appear in 0-1 linear programming problems.

Dietrich and Escudero [3] analyze coefficient improvement for the following special substructure

$$\sum_{j \in J} \{a_j z_j + \sum_{k \in I_j} a_k x_k\} \leq b$$

$$x_k \leq z_j \qquad \forall k \in I_j$$

$$x_k, z_j \in \{0, 1\},$$

i.e., they incorporate variable upper bound constraints $x_k \leq z_j$ in the analysis. Unfortunately, no computational results are reported.

Hoffman and Padberg [6] analyse identification of infeasibility and redundancy as well as coefficient improvement for the following substructure

$$\sum_{j \in J} \sum_{k \in S_j} a_{jk} x_{jk} \leq b$$

$$\sum_{k \in S_j} x_{jk} \leq 1 \qquad \forall j \in J$$

$$x_{jk} \in \{0,1\},$$

i.e., they incorporate non-overlapping clique constraints $\sum_{k \in S_j} x_{jk} \leq 1$ (also called special ordered set constraints) in their analysis. Computational results show an increase in effectiveness at a moderate decrease in efficiency.

2.2 Logical implications

The basic preprocessing and probing techniques can be used effectively to derive logical implications between variables.

First, consider a binary variable x_{k_1}, $k_1 \in B$ and a continuous variable y_k, $k \in C$ and the following extended integer programming problems

$$z_{k_1}^{i_1} = \min \sum_{j \in B^+} a_j^{i_1} x_j - \sum_{j \in B^-} a_j^{i_1} x_j + \sum_{j \in C^+ \setminus \{k\}} g_j^{i_1} y_j - \sum_{j \in C^-} g_j^{i_1} y_j$$

subject to

$$A^{i_1} x + G^{i_1} y \leq b^{i_1}$$

$$x_{k_1} = 1$$

$$l \leq y \leq u$$

$$x \in \{0,1\}^n, y \in RR^m,$$

and

$$z_{k_1}^{i_2} = \max \sum_{j \in B^+} a_j^{i_2} x_j - \sum_{j \in B^-} a_j^{i_2} x_j + \sum_{j \in C^+} g_j^{i_2} y_j - \sum_{j \in C^- \setminus \{k\}} g_j^{i_2} y_j$$

subject to

$$A^{i_2} x + G^{i_2} y \leq b^{i_2}$$

$$x_{k_1} = 1$$

$$l \leq y \leq u$$

$$x \in \{0,1\}^n, y \in RR^m.$$

Clearly, $y_k \leq (b_{i_1} - z_{k_1}^{i_1})/g_k^{i_1}$ and $y_k \geq (z_{k_1}^{i_2} - b_{i_2})/g_k^{i_2}$, i.e., an analysis of the extended system of inequalities may reveal that when $x_{k_1} = 1$, the upper and lower bound on the variable y_k may be improved. In case the improved bounds fix the variable y_k, i.e., $l_k = u_k = v_k$, we have established the logical implication $x_{k_1} = 1 \Rightarrow y_k = v_k$. Analogously, it may be possible to establish a logical implication $x_{k_1} = 0 \Rightarrow y_k = v_k$.

Second, consider two binary variables x_{k_1}, $k_1 \in B$ and x_{k_2}, $k_2 \in B$ and the following extended formulation

$$z = \min \sum_{j \in B^+} a_j^i x_j - \sum_{j \in B^-} a_j^i x_j + \sum_{j \in C^+} g_j^i y_j - \sum_{j \in C^-} g_j^i y_j$$

subject to

$$A^i x + G^i y \leq b^i$$

$$x_{k_1} = 1$$

$$x_{k_2} = 1$$

$$l \leq y \leq u$$

$$x \in \{0,1\}^n, y \in RR^m.$$

If $z > b_i$, then obviously the feasible region of this extended formulation is empty. In that case, the following logical implications have been identified

$$x_{k_1} = 1 \Rightarrow x_{k_2} = 0,$$

$$x_{k_2} = 1 \Rightarrow x_{k_1} = 0.$$

Similarly, the following logical implications can be identified

$$x_{k_1} = 0 \Rightarrow x_{k_2} = 0,$$

$$x_{k_1} = 0 \Rightarrow x_{k_2} = 1,$$

$$x_{k_1} = 1 \Rightarrow x_{k_2} = 1,$$

$$x_{k_2} = 0 \Rightarrow x_{k_1} = 0,$$

$$x_{k_2} = 0 \Rightarrow x_{k_1} = 1,$$

$$x_{k_2} = 1 \Rightarrow x_{k_1} = 1.$$

Implementation
The identification of logical implications proceeds in two phases. Suppose, without loss of generality, that we want to identify logical implications associated with $x_{k_1} = 1$. In the first phase, the system $Ax + Gy \leq b$ is reduced by eliminating variable x_{k_1}, i.e., substituting $x_{k_1} = 1$ throughout. In the second phase, each of the inequalities of the reduced system is analysed in turn, using the basic preprocessing and probing techniques to modify bounds and to fix variables, to establish whether any variables can be fixed. If so, logical implications have been identified.

Logical implications can be used to strengthen various functions embedded in mixed integer optimizers, such as probing, knapsack cover generation, flow cover generation, and primal heuristics.

2.2.1 Enhanced probing techniques

The basic probing techniques tentatively set a binary variable to one of its bounds and explore the logical consequences. Suppose, without loss of generality, that a variable x_k is tentatively set to its upper bound. If there already exist logical implications associated with variable x_k being fixed at its upper bound, they can be effectuated during the search for other logical consequences. Observe that the effectuation of one logical implication may trigger effectuation of various other logical implications.

Note that the system $A^i x + G^i y \leq b^i$ is no longer completely discarded, since existing logical implications may have been established during the analysis of inequalities other than the one currently under consideration. Therefore, fixing variables and improving coefficients of variables is no longer restricted to variables appearing in the constraint under consideration. See for an example of this phenomenon the example below.

Implementation
Suppose, without loss of generality, that we probe with variable x_k set to its upper bound. In the first phase, the system $Ax + Gy \leq b$ is reduced by eliminating *all* the variables that are known to be fixed when the system $Ax + Gy \leq b$ is extended with the equality $x_k = 1$, i.e., all the logical implications that become active when x_k is tentatively fixed at 1 are effectuated. In the second phase, each of the inequalities of the reduced system is analysed in turn, using the basic preprocessing techniques to identify infeasibility and redundancy. If infeasibility is detected, the variable on which we probe can be fixed permanently; if redundancy is detected, the coefficient of the variable on which we probe can be improved in the inequality currently under consideration.

The set of variables that are fixed when the system $Ax + Gy \leq b$ is extended with the equality $x_k = 1$ can be determined efficiently using the logical implications. Consequently, the reduced system can de determined efficiently.

Example
In order to provide some insight into the effectiveness of logical implications, consider the following mixed integer program

$$\min 24x_1 + 12x_2 + 16x_3 + 4y_1 + 2y_2 + 3y_3$$

subject to

$$\begin{aligned} y_1 + 3y_2 &\geq 15 \\ y_1 + 2y_3 &\geq 10 \\ 2y_1 + y_2 &\geq 20 \\ y_1 &\leq 15x_1 \\ y_2 &\leq 20x_2 \\ y_3 &\leq 5x_3 \\ y_1, y_2, y_3 &\in \mathbb{R}_+ \\ x_1, x_2, x_3 &\in \{0, 1\} \end{aligned}$$

Application of the basic preprocessing techniques will set the upper bounds on the continuous variables to 15, 20 and 5 respectively. The logical implications that can be identified from the resulting formulation and their derivations are listed below

Implication		Derivation
$x_1 = 0 \Rightarrow$	$y_1 = 0$	$(x_1 = 0 \Rightarrow y_1 = 0)$
	$y_2 = 20$	$(x_1 = 0 \Rightarrow y_1 = 0 \Rightarrow y_2 = 20)$
	$y_3 = 5$	$(x_1 = 0 \Rightarrow y_1 = 0 \Rightarrow y_3 = 5)$
	$x_2 = 1$	$(x_1 = 0 \Rightarrow y_1 = 0 \Rightarrow y_2 = 20 \Rightarrow x_2 = 1)$
	$x_3 = 1$	$(x_1 = 0 \Rightarrow y_1 = 0 \Rightarrow y_3 = 5 \Rightarrow x_3 = 1)$
$x_2 = 0 \Rightarrow$	$y_2 = 0$	$(x_2 = 0 \Rightarrow y_2 = 0)$
	$y_1 = 15$	$(x_2 = 0 \Rightarrow y_2 = 0 \Rightarrow y_1 = 15)$
	$x_1 = 1$	$(x_2 = 0 \Rightarrow y_2 = 0 \Rightarrow y_1 = 15 \Rightarrow x_1 = 1)$
$x_3 = 0 \Rightarrow$	$y_3 = 0$	$(x_3 = 0 \Rightarrow y_3 = 0)$
	$x_1 = 1$	$(x_3 = 0 \Rightarrow y_3 = 0 \Rightarrow y_1 \geq 10 \Rightarrow x_1 = 1)$

Note that some of these logical implications would not have been identified if the upper bounds on the continuous variables would not have been improved by the basic preprocessing techniques. Also note that $y_3 = 0 \Rightarrow y_1 \geq 10$, although, according to our definition, not a logical implication itself, is used in the derivation of $x_3 = 0 \Rightarrow x_1 = 1$.

Application of the enhanced probing techniques results in the following improved formulation

$$\min 24x_1 + 12x_2 + 16x_3 + 4y_1 + 2y_2 + 3y_3$$

subject to

$$\begin{aligned} 45x_1 + y_1 + 3y_2 &\geq 60 \\ 5x_2 + y_1 + 2y_3 &\geq 15 \\ 10x_2 + 2y_1 + y_2 &\geq 30 \\ y_1 &\leq 15x_1 \\ y_2 &\leq 20x_2 \\ y_3 &\leq 5x_3 \\ 0 \leq y_1 &\leq 15 \\ 0 \leq y_2 &\leq 20 \\ 0 \leq y_3 &\leq 5 \\ x_1, x_2 x_3 &\in \{0, 1\} \end{aligned}$$

The value of the solution of the initial linear programming relaxation is 58.70. The value of the solution of the improved linear programming relaxation is 79.10. The value of the optimal solution is 79.33.

2.2.2 Clique inequalities

Logical implications can be used to derive clique inequalities, i.e., inequalities with only binary variables of the form $\sum_{j \in S^+} x_j - \sum_{j \in S^-} x_j \leq 1 - |S^-|$. Clique inequalities and their theoretical foundation are described in Johnson and Padberg [8]. The construction of clique inequalities is based on logical implications between binary variables. Note that the four types of logical implications between binary variables can be represented as follows:

- $x_i = 1 \Rightarrow x_j = 0$
- $x_i = 1 \Rightarrow \overline{x}_j = 0$
- $\overline{x}_i = 1 \Rightarrow x_j = 0$
- $\overline{x}_i = 1 \Rightarrow \overline{x}_j = 0$

where $\overline{x}_k = 1 - x_k$ denotes the complement of x_k. That is, a logical implication identifies two variables, either original or complemented, that cannot be 1 at the same time in any feasible solution. Construct the graph $G = (B^o \cup B^c, E)$ with B^o the set of original binary variables, B^c the set of complemented binary variables, and E the set of edges, where two variables are joined by an edge if and only if the two variables cannot be 1 at the same time in a feasible solution. Consequently, each implication defines an edge in the graph. Furthermore, there is an edge between each variable and its complement.

It is not hard to see that every (maximal) clique $C = C^o \cup C^c$, with $C^o \subseteq B^o$ and $C^c \subseteq B^c$, defines a valid clique-inequality

$$\sum_{j \in C^o} x_j + \sum_{j \in C^c} \overline{x}_j \leq 1$$

which implies

$$\sum_{j \in C^o} x_j - \sum_{j \in C^c} x_j \leq 1 - |C^c|$$

Two important observations can be made with respect to these clique-inequalities:

- if $|C^o \cap C^c| = 1$ and $k \in C^o \cap C^c$, then $x_j = 0$ for all $j \in C^o \setminus \{k\}$ and $x_j = 1$ for all $j \in C^c \setminus \{k\}$.
- if $|C^o \cap C^c| > 1$ the problem is infeasible.

Example
Consider the following system of inequalities

$$\begin{aligned} 4x_1 + x_2 - 3x_4 &\leq 2 \\ 2x_1 + 3x_2 + 3x_4 &\leq 7 \\ x_1 + 4x_2 + 2x_3 &\leq 5 \\ 3x_1 + x_2 + 5x_3 &\leq 6 \\ x_1, x_2, x_3, x_4 &\in \{0, 1\} \end{aligned}$$

The following implication inequalities will be identified

Implication		Derivation
$x_1 = 1 \Rightarrow$	$x_4 = 1$	$(x_1 = 1 \Rightarrow x_4 = 1)$
	$x_2 = 0$	$(x_1 = 1 \Rightarrow x_4 = 1 \Rightarrow x_2 = 0)$
	$x_3 = 0$	$(x_1 = 1 \Rightarrow x_3 = 0)$
$x_2 = 1 \Rightarrow$	$x_3 = 0$	$(x_2 = 1 \Rightarrow x_3 = 0)$
$x_3 = 1 \Rightarrow$	$x_2 = 0$	$(x_3 = 1 \Rightarrow x_2 = 0)$
	$x_1 = 0$	$(x_3 = 1 \Rightarrow x_1 = 0)$
$x_4 = 0 \Rightarrow$	$x_1 = 0$	$(x_4 = 0 \Rightarrow x_1 = 0)$

The graph $G = (B^o \cup B^c, E)$ for the system of inequalities is given in Figure 1 and contains one maximal clique with cardinality greater than two: $\{x_1, x_2, x_3\}$. This clique defines the inequality $x_1 + x_2 + x_3 \leq 1$.

Figure 1. Auxiliary graph

2.2.3 Implication inequalities

Logical implications define valid inequalities. These inequalities will be referred to as implication inequalities and are given below

- $x_i = 1 \Rightarrow y_j = v_j$ implies $y_j \geq l_j + (v_j - l_j)x_i$.
- $x_i = 1 \Rightarrow y_j = v_j$ implies $y_j \leq u_j - (u_j - v_j)x_i$.
- $x_i = 0 \Rightarrow y_j = v_j$ implies $y_j \geq v_j - (v_j - l_j)x_i$.
- $x_i = 0 \Rightarrow y_j = v_j$ implies $y_j \leq v_j + (u_j - v_j)x_i$.

Automatic disaggregation
A very important consequence of generating implication inequalities is that constraint will automatically be disaggregated. Consider an inequality

$$\sum_{j \in S} y_j \leq (\sum_{j \in S} u_j) x_k,$$

where $y_j \in RR_+$ for all $j \in S$ and $x_k \in \{0, 1\}$. An analysis of this inequality will identify the logical implications $x_k = 0 \Rightarrow y_j = 0$ for all $j \in S$. Each of them defines an implication inequality

$$y_j \leq u_j x_k.$$

Collectively these implication inequalities provide a stronger linear approximation of the feasible region than the original inequality.

Example
To provide a more concrete example, consider the following instance of the capacitated facility location problem (CFLP). There are four possible locations for facilities and three demand points. If a facility is opened in one of the four locations, it will have a production capacity of 100 units. The costs associated with opening a facility at a certain location are 400, 500, 300, and 150 respectively; the demand points require 80, 70, 40 units respectively.

The mixed integer program for this instance of CFLP is given by

$$\begin{aligned} \min \quad & 400x_1 + 500x_2 + 300x_3 + 150x_4 + \\ & 20y_{11} + 48y_{21} + 26y_{31} + 24y_{41} + \\ & 40y_{12} + 15y_{22} + 35y_{32} + 50y_{42} + \\ & 50y_{13} + 26y_{23} + 18y_{33} + 35y_{43} \end{aligned}$$

subject to

$$\begin{aligned} y_{11} + y_{21} + y_{31} + y_{41} &= 80 \\ y_{12} + y_{22} + y_{32} + y_{42} &= 70 \\ y_{13} + y_{23} + y_{33} + y_{43} &= 40 \\ y_{11} + y_{12} + y_{13} &\leq 100x_1 \\ y_{21} + y_{22} + y_{23} &\leq 100x_2 \end{aligned}$$

$$y_{31} + y_{32} + y_{33} \leq 100x_3$$
$$y_{41} + y_{42} + y_{43} \leq 100x_4$$
$$y_{ij} \in \mathbb{R}_+ \qquad x_i \in \{0,1\}.$$

Application of the basic preprocessing techniques will improve the upper bounds on the continuous variables to $y_{ij} \leq d_j$, where d_j indicates the demand associated with point j. The following implication inequalities will be identified

$$y_{ij} \leq d_j x_j$$

The solution to the initial linear programming relaxation is given by $y_{11} = 80, x_1 = 0.8, y_{22} = 70, x_2 = 0.7, y_{33} = 40, x_3 = 0.4$ and all other variables equal to 0. Obviously, the solution violates the three implication inequalities

$$y_{11} \leq 80x_1$$
$$y_{22} \leq 70x_2$$
$$y_{33} \leq 40x_3$$

If these three implication inequalities are added to the formulation and the thus obtained linear program is solved, the solution is integral and therefore also optimal.

Observe that the clique inequalities as well as implication inequalities are globally valid and can thus be used effectively in a branch-and-cut approach, i.e., at every node of the branch-and-bound tree. Furthermore, these inequalities may also strengthen various other components of mixed integer optimizers. The variable upper bound constraints, that have been identified as implication inequalities, and the clique constraints may strengthen the preprocessing techniques based on special substructures (such as the ones studied by Dietrich and Escudero [3] and Hoffman and Padberg [6]). Variable upper bound constraints are also of crucial importance for the generation of simple and extended generalized flow cover inequalities as done by systems such as MPSARX [11] and MINTO [12].

2.2.4 Elimination of variables

Logical implications associated with fixing a binary variable at 1 together with logical implications associated with fixing that variable at 0 may even reveal equalities, which may be used to eliminate variables:

- $x_i = 0 \Rightarrow y_j = u_j$ and $x_i = 1 \Rightarrow y_j = v_j$ imply $y_j = v_j$.
- $x_i = 0 \Rightarrow y_j = l_j$ and $x_i = 1 \Rightarrow y_j = u_j$ imply $y_j = l_j + (u_j - l_j)x_i$.
- $x_i = 0 \Rightarrow y_j = u_j$ and $x_i = 1 \Rightarrow y_j = l_j$ imply $y_j = u_j - (u_j - l_j)x_i$.

2.3 Probing on constraints

The probing techniques exploit the fact that in any feasible solution the value of a binary variable is either 0 or 1. Or, formulated slightly different, the probing techniques exploit the fact that there exists a simple scheme to divide the solution space in two parts. This observation suggests that in certain situations it is also possible to probe on constraints, as opposed to probing on variables.

Consider a clique inequality $\sum_{j \in N^+} x_j - \sum_{j \in N^-} x_j \leq 1 - |N^-|$. The value of the left hand side will be either $-|N^-|$ or $1 - |N^-|$. If it is $-|N^-|$, then $x_j = 0$ for all $j \in N^+$ and $x_j = 1$ for all $j \in N^-$. If it is $1 - |N^-|$, then either $x_k = 1$ for some $k \in N^+$, $x_j = 0$ for all $j \in N^+ \setminus \{k\}$, and $x_j = 1$ for all $j \in N^-$, or $x_j = 0$ for all $j \in N^+$, $x_k = 0$ for some $k \in N^-$, and $x_j = 1$ for all $j \in N^- \setminus \{k\}$.

Generalized variable fixing
Instead of trying to fix a single binary variable, we can try to fix all the variables in a clique inequality $\sum_{j \in N^+} x_j - \sum_{j \in N^-} x_j \leq 1 - |N^-|$. Consider the following extended integer programming problem

$$z = \min \sum_{j \in B^+} a^i_j x_j - \sum_{j \in B^-} a^i_j x_j + \sum_{j \in C^+} a^i_j y_j - \sum_{j \in C^-} a^i_j y_j$$

subject to

$$A^i x + G^i y \leq b^i$$

$$\sum_{j \in N^+} x_j - \sum_{j \in N^-} x_j = 1 - |N^-|$$

$$x \in \{0,1\}^n, y \in RR^m$$

If $z > b_i$, then $\sum_{j \in N^+} x_j - \sum_{j \in N^-} x_j \neq 1 - |N^-|$ in any feasible solution, i.e., $\sum_{j \in N^+} x_j - \sum_{j \in N^-} x_j = -|N^-|$ and thus $x_j = 0$ for all $j \in N^+$ and $x_j = 1$ for all $j \in N^-$.

Generalized coefficient reduction
Consider a clique inequality $\sum_{j \in N^+} x_j - \sum_{j \in N^-} x_j \leq 1 - |N^-|$, with $N^+ \subseteq B^+$ and $N^- \subseteq B^-$, $|N^-| \leq 1$ and the following extended integer programming problem

$$z = \max \sum_{j \in B^+} a^i_j x_j - \sum_{j \in B^-} a^i_j x_j + \sum_{j \in C^+} a^i_j y_j - \sum_{j \in C^-} a^i_j y_j$$

subject to

$$A^i x + G^i y \leq b^i$$

$$\sum_{j\in N^+} x_j - \sum_{j\in N^-} x_j = -|N^-|$$

$$x \in \{0,1\}^n, y \in RR^m.$$

If $z < b_i$, then a^i_j may be decreased for all $j \in N^+$, a^i_j may be increased for all $j \in N^-$, and b_i may be decreased by $b_i - z$ without changing the set of feasible solutions.

In case $\sum_{j\in N^+} x_j - \sum_{j\in N^-} x_j = 1 - |N^-|$, then the inequality $a^i x + g^i y \leq b_i$ can be rewritten as

$$\sum_{j\in B^+\setminus N^+} a^i_j x_j + \sum_{j\in N^+} (a^i_j - \delta)x_j - \sum_{j\in B^-\setminus N^-} a^i_j x_j - \sum_{j\in N^+} (a^i_j - \delta)x_j +$$

$$\sum_{j\in C^+} a^i_j y_j - \sum_{j\in C^-} a^i_j y_j \leq b_i - \delta + |N^-|\delta \qquad \delta \in RR_+.$$

Therefore, also under the assumption that $\sum_{j\in N^+} x_j - \sum_{j\in N^-} x_j = 1 - |N^-|$, a^i_j may be decreased for all $j \in N^+$, a^i_j may be increased for all $j \in N^-$, and b_i may be decreased by $b_i - z$ without changing the set of feasible solutions.

3 Computational results

The basic preprocessing and probing techniques as well as the various applications of logical implications described in the previous sections have been incorporated in MINTO, a Mixed INTeger Optimizer [12]. MINTO is a software system that solves mixed integer programs by a linear programming based branch-and-bound algorithm. Moreover, the user can enrich the basic algorithm by providing a variety of specialized application routines that can customize MINTO to achieve maximum efficiency for a problem class.

We want to emphasize that there are many other techniques, besides preprocessing and probing, that modify a given representation of a mixed integer programming problem in order to reduce the set of feasible solutions of the linear programming relaxation. Various constraint generation techniques, such as knapsack cover generation and flow cover generation, have proven to be quite effective. In addition to the techniques aiming at improving the linear programming relaxation, powerful mixed integer optimizers should also perform reduced cost fixing, have primal heuristics, and different branching strategies. In our computational study, we have concentrated on preprocessing and probing techniques. Our results indicate that these techniques are quite effective in reducing the integrality gap and the size of the branch-and-bound tree. Consequently, they should be incorporated in any mixed integer optimizer.

The effectiveness of the preprocessing and probing techniques has been tested on a set of 10 mixed integer programming problems. Table 1 shows for each of these problems its name, the number of constraints, variables, and nonzero coefficients, and the number of continuous, binary and integer variables.

Problem	#c	#v	#nz	#c	#b	#i
egout	98	141	282	86	55	0
fixnet3	478	878	1756	500	378	0
fixnet4	478	878	1756	500	378	0
fixnet6	478	878	1756	500	378	0
khb05250	101	1350	2700	1326	24	0
gen	780	870	2592	720	144	6
att	1260	1112	4986	280	832	0
sample2	45	67	147	46	21	0
p0033	15	33	98	0	33	0
lseu	28	89	309	0	89	0

Table 1: Characteristics of the test problems

Modifying the representation of a mixed integer program in MINTO involves four phases. In the first phase, the basic preprocessing and probing techniques are applied. Every row of the constraint matrix is anlyzed to identify infeasibility or redundancy, to improve coefficients and bounds, and to fix variables. If, after processing every row in the constraint matrix, some variables have been fixed or some bounds have been improved, the process is repeated. In the second phase, logical implications are identified and enhanced probing techniques are applied. Every binary variable is analyzed by temporarily fixing it at one of its bounds, effectuating all possible logical implications, and analyzing every row in the reduced system using the basic preprocessing and probing techniques to identify logical implications, improve coefficients, and fix variables. If, after processing every binary variable, some logical implications have been identified or some variables have been fixed, the process is repeated. In the third phase, the logical implications that have been identified are used to construct the implication graph and the resulting implication graph is analyzed to identify clique inequalities. In the last phase, implication and clique inequalities are generated 'on the fly', i.e., they are iteratively added to the linear program in case they are violated by the current linear programming relaxation.

Table 2 shows the value of the linear program relaxation (z_{LP}), the value of the linear programming relaxation after the basic preprocessing and probing techniques have been applied (z^1_{LP}), the value of the linear programming relaxation after the basic preprocessing and probing techniques as well as the enhanced probing techniques have been applied (z^2_{LP}), the value of the linear programming relaxation after the basic preprocessing and probing techniques and the enhanced probing techniques have been applied and the derived clique and implication inequalities have been added (z^3_{LP}), and, finally, the value of the mixed integer program (z_{MIP}).

Problem	z_{LP}	z^1_{LP}	z^2_{LP}	z^3_{LP}	z_{MIP}
egout	149.588	495.565	511.875	562.112	568.100
fixnet3	40717.0	49051.5	50413.7	50414.2	51973.0
fixnet4	4257.96	6879.21	7703.47	7703.47	8936.00
fixnet6	1200.88	2527.43	3192.04	3192.53	3983.00
khb05250	95919464.	95919464.	95919464.	106750366.	106940226.
gen	112130.	112233.	112271.	112271.	112313.
att	125.969	149.052	149.145	149.145	160.200
sample2	247.000	247.000	247.000	290.480	375.000
p0033	2520.81	2828.33	2828.33	2838.54	3089.00
lseu	834.682	947.957	947.957	947.957	1120.00

Table 2: Effect of preprocessing and probing techniques on LP value

Table 3 shows the number of nodes required to solve the problem to optimality when MINTO is used without any preprocessing and probing (option -p0), when MINTO is used with only basic preprocessing and probing (option -p1), and when MINTO is used in its default setting, i.e., with both basic and enhanced preprocessing and probing.

Inspecting the computational results, we observe the following. First, the preprocessing and probing techniques are quite effective in reducing the integrality gap and the overall effort required to solve most of these problems. Second, activating preprocessing and probing does not always lead to a smaller search tree. This shows, that at this moment, we do not have a clear understanding of how the different techniques embedded in state-of-the-art mixed integer optimizers interact.

4 Conclusion

In this paper, we have given an overview of simple and advanced preprocessing and probing techniques to improve the representation of a mixed integer program. Our computational results have demonstrated the effectiveness of these techniques.

Problem	minto -p0	minto -p1	minto
egout	553	9	3
fixnet3	131	5	5
fixnet4	2561	2739	1031
fixnet6	4795	3977	4305
khb05250	11483	63	13
gen	11	19	15
att	6459	133	127
sample2	336	71	51
p0033	15	7	7
lseu	297	464	464

Table 3: Effect of preprocessing and probing techniques on branch-and-bound tree

As indicated above, these techniques can be used to enhance the performance of a mixed integer optimizer. However, in a powerful mixed integer optimizer they should be used in conjunction with other techniques, such as knapsack cover generation, flow cover generation, and primal heuristics.

References

[1] A.L. Brearley, G. Mitra, H.P. Williams, 1975. Analysis of mathematical programming problems prior to applying the simplex algorithm. *Math. Prog. 8*, 54-83.

[2] H. Crowder, E.L. Johnson, M.W. Padberg, 1983. Solving large-scale zero-one linear programming problems. *Oper. Res. 31*, 803-834.

[3] B.L. Dietrich, L.F. Escudero, 1990. Coefficient reduction for knapsack-like constraints in 0-1 program with variable upper bounds. *Oper. Res. Letters 2*, 9-14.

[4] M. Guignard, K. Spielberg, 1981. Logical reduction methods in zero-one programming. *Oper. Res. 29*, 49-74.

[5] K.L. Hoffman, M. Padberg, 1985) LP-based combinatorial problem solving. *Annals of Oper. Res. 4*, 145-194.

[6] K.L. Hoffman, M. Padberg, 1991. Improving LP-representation of zero-one linear programs for branch-and-cut. *ORSA J. Comput. 3*, 121-134.

[7] E.L. Johnson, M.M. Kostreva, U. Suhl, 1981. Solving 0-1 integer programming problems arising from large-scale planning models. *Oper. Res. 33*, 803-819.

[8] E.L. Johnson, M. Padberg, 1982. Degree-two inequalities, clique facets and biperfect graphs. *Annals of Discrete Mathematics 16*, 169-187.

[9] E.L. Johnson, 1989. Modeling and strong linear programs for mixed integer programming. S.E. Wallace (ed.). *Algorithms and Model Formulation in Mathematical Programming*, NATO ASI Series F51, Heidelberg.

[10] M. Padberg, G. Rinaldi, 1991. A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM Review 33*, 60-100.

[11] T. van Roy, L.A. Wolsey, 1987. Solving mixed 0-1 programs by automatic reformulation. *Oper. Res. 35*, 45-57.

[12] G.L Nemhauser, M.W.P. Savelsbergh, G.C. Sigismondi, 1994. MINTO, a Mixed INTeger Optimizer. *Oper. Res. Letters*, to appear.