Approximating the Stability Region for Binary Mixed-Integer Programs

Fatma Kılınç-Karzan, Alejandro Toriello,
Shabbir Ahmed*, George Nemhauser, Martin Savelsbergh

March 8, 2009

Abstract

We consider optimization problems with some binary variables, where the objective function is linear in these variables. The *stability region* of a given solution is the polyhedral set of objective coefficients for which the solution is optimal. Knowledge of this set provides valuable information for sensitivity analysis and re-optimization. An exact description of the stability region may require an exponential number of inequalities. We develop polyhedral inner and outer approximations of linear size. Furthermore, when a new objective function is not in the stability region, we produce a list of high-quality solutions that can be used as a quick heuristic or as a warm start for future solves.

1 Introduction

Suppose we have an NP-hard optimization problem with some binary variables and a linear objective function for these variables. In addition to finding an optimal solution, we would like to know how much the objective vector can change while maintaining the optimality of the solution. This stability information is useful, for example, when solving problems with perturbed objective functions rapidly.

We consider the optimization problem

$$\begin{aligned} z^* = \max\ & c^*x + h(y) \\ \text{s.t. } & (x,y) \in X \\ & x \in \{0,1\}^n, \end{aligned} \tag{$P(c^*)$}$$

where $c^*x = \sum_{i\in N} c_i^* x_i$, $N = \{1,\dots,n\}$ and $h : \text{proj}_y(X) \to \mathbb{R}$. We do not give specific requirements on h and X, but we assume that an optimal solution (x^*, y^*) exists and can be computed. We are interested in the stability of (x^*, y^*) with respect to variations of the cost vector c^*. By possibly complementing variables, we assume without loss of generality that $x^* = 0$, so the optimal value of $P(c^*)$ is $z^* = h(y^*)$.

Definition 1.1. The *stability region* [10] of (x^*, y^*) is the region $C \subseteq \mathbb{R}^n$ s.t. $c \in C$ if and only if (x^*, y^*) is optimal for $P(c)$. That is, $C = \{c \in \mathbb{R}^n : c(x - x^*) \le h(y^*) - h(y), \forall\ (x,y) \in X\}$.

*Corresponding author. E-mail address:sahmed@isye.gatech.edu.

A familiar example of a stability region comes from linear optimization. In a linear program, the stability region of a solution is the cone generated by the set of constraints that are binding at the solution.

Stability analysis is related to sensitivity analysis, and some previous related work exists in this field. Whereas our approach studies a general optimization model with binary variables, other authors have focused on specific problems, such as shortest paths and maximum capacity paths [8], assignment problems [6], minimum spanning trees and shortest path trees [11], and knapsacks [4]. In addition, right-hand side vector sensitivity in integer and mixed-integer linear programs is studied in [2, 5, 9].

Remark 1.1. *Let $(\hat{x}, \hat{y})$ be feasible for $P(c^*)$, with objective value $\hat{z} = c^*\hat{x} + h(\hat{y}) \leq z^*$. Suppose $\hat{c} \in \mathbb{R}^n$ satisfies $\hat{c}\hat{x} > z^* - \hat{z} + c^*\hat{x}$. Then $\hat{c} \notin C$.*

Remark 1.2. *Let $\hat{x} \in \{0,1\}^n$, and define $v(\hat{x}) = \max_{(\hat{x},y)\in X} h(y)$, with $v(\hat{x}) = -\infty$ if no y satisfies $(\hat{x}, y) \in X$. Then $C = \{c \in \mathbb{R}^n : cx \leq h(y^*) - v(x), \forall\ x \in \{0,1\}^n\}$.*

Remark 1.2 implies that C is a polyhedron possibly defined by an exponential number of inequalities. Perhaps because of the relative difficulty involved in working with exponentially many inequalities, stability regions do not appear in the literature (to our knowledge) as often as an associated concept, the stability radius. The *stability radius* of (x^*, y^*) with respect to $P(c^*)$ is defined as

$$\rho^* = \sup\{\rho : (x^*, y^*) \text{ is optimal for } P(c), \forall\ c : \|c - c^*\|_\infty \leq \rho\}, \tag{1}$$

where $\|d\|_\infty = \max_{i\in N}\{|d_i|\}$. Within stability analysis, much attention has been devoted to computing or approximating ρ^*, and to the complexity of such a task [3, 10]. However, the $\|\cdot\|_\infty$-ball associated with ρ^* may drastically underapproximate C. (Consider, for example, the case when $P(c^*)$ has an alternative optimal solution that differs from (x^*, y^*) in one binary variable.) We choose instead to approximate C using polyhedra defined by polynomially many inequalities. The next two definitions further explain this approximation.

Definition 1.2. Let C be the stability region of (x^*, y^*). An *outer approximation* of C is a region $C^+ \subseteq \mathbb{R}^n$ satisfying $C^+ \supseteq C$. An *inner approximation* of C is a region $C^- \subseteq \mathbb{R}^n$ satisfying $C^- \subseteq C$.

Let $(\hat{x}, \hat{y})$ be feasible for $P(c^*)$. By Remark 1.1, the set $\{c \in \mathbb{R}^n : c\hat{x} \leq z^* - h(\hat{y})\}$ is an outer approximation of C. The set $\{c \in \mathbb{R}^n : c \leq c^*\}$ is a trivial inner approximation of C. The following are two simple but important consequences of Definition 1.2 that help us obtain small outer approximations and large inner approximations.

Proposition 1.3.

i) If C_1^+, C_2^+ are outer approximations, $C_1^+ \cap C_2^+$ is an outer approximation.

ii) If C_1^-, C_2^- are inner approximations, $\text{conv}(C_1^- \cup C_2^-)$ is an inner approximation.

The rest of the paper proceeds as follows. Section 2 explains how to approximate the stability region by solving problems related to $P(c^*)$. Section 3 expresses the results of the previous section as an algorithm. Section 4 evaluates the quality of the approximations via some computational experiments. Section 5 gives conclusions.

2 Approximating the Stability Region

In this section, we show how to obtain inner and outer approximations of C by solving n problems $\{P_j : j \in N\}$, where each P_j is given by

$$\begin{aligned} z_j = \max\ & c^*x + h(y) \\ \text{s.t. } & (x, y) \in X \\ & x \in \{0,1\}^n \\ & x_j = 1. \end{aligned} \tag{P_j}$$

Throughout this section, we assume without loss of generality that all problems P_j are feasible. If P_j is infeasible for some $j \in N$, then every feasible solution to $P(c^*)$ has $x_j = 0$. Thus, c_j cannot in any way affect optimality, and we can ignore it. Accordingly, we assume that we have solved the problems P_j for every $j \in N$, and determined optimal solutions (x^j, y^j) with corresponding objective values z_j. An important question is whether solving the problems P_j is necessary to obtain the z_j values, or if a more efficient method exists. This is especially pertinent if $P(c^*)$ is NP-hard, because each P_j is then NP-hard as well. However, Van Hoesel and Wagelmans [12] have shown that if $P(c^*)$ is NP-hard, then finding z_j is also NP-hard.

The following observation follows directly from Remark 1.1 and Proposition 1.3.

Proposition 2.1. *The set* $C^+ = \left\{c \in \mathbb{R}^n : cx^j \le z^* - z_j + c^*x^j,\ \forall\ j \in N\right\}$ *is an outer approximation of* C.

Let $\gamma_j = z^* - z_j + c^*_j$. Observe that the outer approximation C^+ of Proposition 2.1 satisfies

$$\{c \in C^+ : c \ge c^*\} \subseteq \{c \in \mathbb{R}^n : c^*_j \le c_j \le \gamma_j,\ \forall\ j \in N\}. \tag{2}$$

In other words, in the direction $c \ge c^*$, the stability region C of (x^*, y^*) is contained in a box defined by the values γ_j.

To determine an inner approximation, we make use of the next result.

Proposition 2.2. *Let* $\tilde{c}^j = (c^*_1, \ldots, c^*_{j-1}, \gamma_j, c^*_{j+1}, \ldots, c^*_n)$. *Then* $\tilde{c}^j \in C,\ \forall\ j \in N$.

Proof. Observe that the objective value of (x^*, y^*) in $P(\tilde{c}^j)$ is $\tilde{c}^j x^* + h(y^*) = z^*$. Let $(\hat{x}, \hat{y})$ be feasible for $P(\tilde{c}^j)$. If $\hat{x}_j = 0$, then $\tilde{c}^j\hat{x} + h(\hat{y}) = c^*\hat{x} + h(\hat{y}) = \hat{z} \le z^*$ by assumption. If $\hat{x}_j = 1$, then $\tilde{c}^j\hat{x} + h(\hat{y}) = c^*\hat{x} + h(\hat{y}) - z_j + z^* = \hat{z} - z_j + z^* \le z^*$, where the last inequality follows because $(\hat{x}, \hat{y})$ is feasible for P_j. □

Theorem 2.3. *Suppose we order the* x *variables so that* $z^* \ge z_1 \ge z_2 \ge \cdots \ge z_n$ *holds. Then*

$$C^- = \left\{c \ge c^* : \sum_{i=1}^{j} c_i \le \gamma_j + \sum_{i=1}^{j-1} c^*_i,\ \forall\ j \in N\right\} \tag{3}$$

is an inner approximation of C.

Proof. For any $x \in \{0,1\}^n$, we do not know the exact value of the right-hand side in the inequality $cx \le h(y^*) - v(x)$. However, Proposition 2.2 states that the points $\{\tilde{c}^j\}_{j \in N}$ lie in the stability region of (x^*, y^*). Therefore, the halfspace defined by each inequality should contain these n points. Thus, the set

$$\tilde{C}^- = \left\{c \in \mathbb{R}^n : \sum_{i \in I} c_i \le \max\left\{\sum_{i \in I} \tilde{c}^j_i : j \in N\right\}, \forall\ \emptyset \ne I \subseteq N\right\}$$

is an inner approximation of C. Now let $\emptyset \neq I \subseteq N$. We must prove that the inequality corresponding to I in $\tilde{C}^-$ is dominated by the inequalities defining C^-, $\forall\ c \geq c^*$.

Claim. Let $t \in N$ be the smallest index satisfying $I \subseteq \{1, \ldots, t\}$. Then t satisfies

$$\max\left\{\sum_{i\in I} \tilde{c}_i^j : j \in N\right\} = \sum_{i\in I} \tilde{c}_i^t.$$

Proof of claim. Let $s \in N$. If $s > t$, we have

$$\sum_{i\in I} \tilde{c}_i^s = \sum_{i\in I} c_i^* \leq \underbrace{z^* - z_t}_{\geq 0} + \sum_{i\in I} c_i^* = \sum_{i\in I} \tilde{c}_i^t.$$

Similarly, if $s \leq t$, we get

$$\sum_{i\in I} \tilde{c}_i^s \leq z^* - \underbrace{z_s}_{\geq z_t} + \sum_{i\in I} c_i^* \leq z^* - z_t + \sum_{i\in I} c_i^* = \sum_{i\in I} \tilde{c}_i^t. \qquad \square$$

We now apply the claim to the following inequality, which is included in the definition of C^-:

$$\sum_{i=1}^{t} c_i \leq \gamma_t + \sum_{i=1}^{t-1} c_i^* = z^* - z_t + \sum_{i=1}^{t} c_i^*.$$

Rearranging terms yields

$$\begin{aligned}\sum_{i\in I} c_i &\leq z^* - z_t + \sum_{i\in I} c_i^* + \underbrace{\sum_{i\in\{1,\ldots,t\}\setminus I} (c_i^* - c_i)}_{\leq 0} \\ &\leq z^* - z_t + \sum_{i\in I} c_i^* = \max\left\{\sum_{i\in I} \tilde{c}_i^j : j \in N\right\}. \qquad \square\end{aligned}$$

Corollary 2.4. *The set $\{c + d : c \in C^-, d \leq 0\}$ is an inner approximation of C.*

Proof. Note that if (x^*, y^*) (with $x^* = 0$) is optimal for $P(c^*)$, then the set $\{c \in \mathbb{R}^n : c \leq c^*\}$ is a trivial inner approximation of C. Using Proposition 1.3, we apply this observation to C^-. $\square$

These last two results motivate a natural algorithm for determining an inner approximation. Solve each of the problems P_j in turn, sort them by objective value, and compute the inner approximation as indicated in Theorem 2.3. As we shall see in the next section, this procedure can be modified slightly to potentially reduce the number of solves.

3 The Algorithm

Algorithm 1 begins with a problem of the form $P(c^*)$ and an optimal solution (x^*, y^*) with $x^* = 0$. It then adds a cut $\sum_{i\in N} x_i \geq 1$, forcing at least one of the x variables to one. After resolving, it determines which of the x variables switched, removes their coefficients from the cut, and repeats. The end result is a list of solutions, ordered by objective value, which covers all possible x variables. (Variables not covered by any solution on the list are those fixed to zero in any feasible solution.)

Algorithm 1 Binary Solution Cover

Require: An optimization problem $P(c^*)$ with optimal solution (x^*, y^*) satisfying $x^* = 0$.

Set $(x^0, y^0) \leftarrow (x^*, y^*)$, $z_0 \leftarrow z^*$, $I \leftarrow N$, $I_\infty \leftarrow \emptyset$.
Add cut $D \equiv (\sum_{i \in I} x_i \geq 1)$ to $P(c^*)$.
for $k = 1, \ldots, n$ **do**
  Resolve the modified $P(c^*)$; get new optimal solution (x^k, y^k) and objective value $c^* x^k + h(y^k) = z_k$.
  if $P(c^*)$ is infeasible **then**
    Set $I_\infty \leftarrow I$.
    Return k and exit.
  end if
  Set $I_k^+ \leftarrow \{i \in N : x_i^k = 1\}$, $I_k^- \leftarrow I \cap I_k^+$.
  Set $I \leftarrow I \setminus I_k^-$; modify cut D accordingly.
end for

For future reference, we formally define the relevant information gathered during the execution of Algorithm 1 as follows. The solution in the k-th iteration is denoted by (x^k, y^k) and has objective function value z_k. The set $I_k^+ \subseteq N$ indicates which variables have value one in (x^k, y^k). The set $I_k^- \subseteq I_k^+$ indicates which of these variables have value one for the first time.

An outer approximation is easily obtained applying Remark 1.1 to each solution (x^k, y^k). To determine an inner approximation, we must first establish a preliminary fact.

Proposition 3.1. *Let $i \in I_k^-$, for some k. Then (x^k, y^k) is optimal for P_i.*

Proof. Suppose not, and let $(\hat{x}, \hat{y})$ be optimal for P_i with $\hat{z} > z_k$. At the beginning of iteration k, we have $i \in I$, implying that no solution so far has covered i. But then, since $(\hat{x}, \hat{y})$ is feasible for P_i, we have $\hat{x}_i = 1$. Furthermore, since $i \in I$, we also have $\sum_{i \in I} \hat{x}_i \geq 1$. This means $(\hat{x}, \hat{y})$ is feasible for the modified $P(c^*)$ in the current iteration, contradicting the optimality of (x^k, y^k). □

Note that (x^k, y^k) is not necessarily optimal for P_j, for $j \in I_k^+ \setminus I_k^-$. We now combine Proposition 3.1 with Theorem 2.3 to construct our inner approximation.

Theorem 3.2. *Suppose Algorithm 1 is run on problem $P(c^*)$, terminating after $\ell \leq n$ steps. Let $(x^k, y^k), z_k, I_k^+, I_k^-, \forall\ k = 1, \ldots, \ell$ and I_∞ be obtained from the algorithm. Then*

$$C^- = \left\{ c \geq c^* : \sum_{i \in I_1^- \cup \cdots \cup I_k^-} c_i \leq z^* - z_k + \sum_{i \in I_1^- \cup \cdots \cup I_k^-} c_i^*,\ \forall\ k = 1, \ldots, \ell \right\} \tag{4}$$

is an inner approximation of C.

We can also apply Corollary 2.4 to get an inner approximation that is not restricted to values greater than or equal to c^*.

It is important to note that the stability region C depends only on (x^*, y^*) and $h(y)$, and not on c^*. However, the inner approximation we calculate using Algorithm 1 does depend on c^*, because c^* appears in the right-hand side of each inequality and also because different starting costs may

determine different solutions in the algorithm when we add the inequality $\sum_{i\in I} x_i \geq 1$. So any $c^* \in C$ can be used in Algorithm 1 to obtain a possibly different inner approximation.

An interesting question is whether we can obtain additional information by running the algorithm on a bigger variable set. Specifically, let $M \subseteq N$ and suppose we are interested in the stability region with respect to the binary variables only in the index set M. Proposition 3.3 essentially states that applying the algorithm to the larger set N does not yield better results.

Proposition 3.3. *Suppose a run of Algorithm 1 on N produces the solution list $\{(x^k, y^k)\}_{k=1,\ldots,\ell}$ and the inner approximation C_N^-. Let $J_k^- = I_k^- \cap M, \forall\ k = 1, \ldots, \ell$. Then the list $\{x^k, y^k\}_{k:J_k^- \neq \emptyset}$ is a list of solutions satisfying the conditions of Algorithm 1 for M, and the corresponding inner approximation C_M^- satisfies*

$$C_M^- \times \{c_i = c_i^*,\ \forall\ i \in N \setminus M\} = C_N^- \cap \{c \in \mathbb{R}^n : c_i = c_i^*,\ \forall\ i \in N \setminus M\}.$$

Proof. For the first part, note that the list $\{x^k, y^k\}_{k:J_k^- \neq \emptyset}$ is simply a restriction of the original list to the variables indexed by M. Since it was generated by Algorithm 1, the list is ordered by objective function value, and each solution (x^k, y^k) is optimal for $P_j, \forall\ j \in J_k^-$.

For the second part, we have

$$\begin{aligned}
&C_N^- \cap \{c : c_i = c_i^*,\ \forall\ i \in N \setminus M\} \\
&= \left\{ c \geq c^* : \sum_{i \in J_1^- \cup \cdots \cup J_k^-} c_i \leq z^* - z_k + \sum_{i \in J_1^- \cup \cdots \cup J_k^-} c_i^*,\ \forall\ k = 1, \ldots, \ell; c_i = c_i^*,\ \forall\ i \in N \setminus M \right\} \\
&= \left\{ c \geq c^* : \sum_{i \in J_1^- \cup \cdots \cup J_k^-} c_i \leq z^* - z_k + \sum_{i \in J_1^- \cup \cdots \cup J_k^-} c_i^*,\ \forall\ k \text{ with } J_k^- \neq \emptyset; c_i = c_i^*,\ \forall\ i \in N \setminus M \right\},
\end{aligned}$$

where the last set equality follows because the inequalities defined for k with $J_k^- = \emptyset$ are dominated by previous inequalities, since the z_i values are non-decreasing. Restricting this last set to the cost coefficients indexed by M yields C_M^-. □

Proposition 3.3 does not guarantee that running Algorithm 1 on N and then restricting C_N^- to obtain an inner approximation for M would yield the same approximation as running the algorithm directly on M. The result assumes a particular list of solutions is used to construct both approximations, but it is possible to obtain a different list for each approximation.

4 Computational Results

We next address the quality of approximation of the inner and outer regions C^-, C^+ generated by Algorithm 1. For any $c \in C^+ \setminus C^-$, we are unable to determine the optimality of (x^*, y^*) for $P(c)$ without re-optimizing. Ideally, we would like to estimate the volume of this uncertainty region $C^+ \setminus C^-$, and perhaps compare it to the volume of C^- and C^+. However, even for problems with modest dimension we cannot efficiently estimate this volume. And although volume computation and general integration by sophisticated random walk techniques is an active area of research [7, 13], no practical volume computation algorithm has yet been developed.

In light of these computational difficulties, we have developed a "shooting" experiment to give some idea of the relative sizes of C^-, C^+ and $C^+ \setminus C^-$. Starting from the original objective vector

c^*, we generate a random direction d by uniformly sampling each component from $[0, 1]$. We then compute the quantities

$$\lambda^- = \max_\lambda \left\{ c^* + \lambda \frac{d}{\|d\|} \in C^- \right\} \qquad \lambda^+ = \max_\lambda \left\{ c^* + \lambda \frac{d}{\|d\|} \in C^+ \right\}, \tag{5}$$

and use the values to compare the relative sizes of C^- and C^+ in the direction d.

We performed the shooting experiment on the problem instances contained in MIPLIB 3.0 [1]. For a given instance, we considered as x variables all binary variables that satisfied the following conditions:

- The variable is not fixed to its optimal value in all feasible solutions. In terms of Algorithm 1, this means the variable does not belong to the set I_∞.
- There is no alternate optimal solution with the variable set to its complement. That is, we have $z^* > z_j$.

We refer to such binary variables as "active." If a particular MIPLIB instance did not have any active binary variables, we discarded it. We also skipped problems with more than 5,000 binary variables and problems which did not solve to optimality within one hour of CPU time. All computational experiments were carried out on a system with two 2.4 GHz Xeon processors and 2 GB RAM, and using CPLEX 11.1 (with default settings) as the optimization engine. For each instance, we generated 100,000 direction vectors d.

Table 1 contains average results for our experiments. For each instance, we report the total number of decision variables (# vars), the total number of binary variables (# bin), the number of active binary variables (# active), the Euclidean norm of the cost vector corresponding to the active variables ($\|c^*\|$), the average λ^- and λ^+ values, and their ratio. For example, for problem `lseu`, 85 of 89 decision variables are active. In the directions tested, C^- occupies 87% of the volume of C^+, but both regions allow only for a relatively small change in c^*, since $\|c^*\|$ is significantly larger than λ^- and λ^+.

As Table 1 indicates, the estimated volume ratio of C^- and C^+ varies significantly from one instance to the next. On one end, Algorithm 1 delivers a fairly tight approximation for problems `dcmulti` (92%), `enigma` (95%), and `lseu` (87%), to name a few. In fact, Algorithm 1 even delivers the exact stability region for `p0033`. On the other, the volume ratio is very small on instances such as `p2756` (7%), `vpm1` (3%) and `vpm2` (14%). This discrepancy is certainly in part due to differences in the number of active binary variables (14 in `enigma` and 2,391 in `p2756`,) since volume differences tend to grow as dimension grows. However, part of this discrepancy is also instance dependent, as some problems with similar numbers of active variables have very different ratios.

Overall, the experimental results indicate that the volume of $C^+ \setminus C^-$ is significant for some instances. Therefore, we next address the cause of this discrepancy: Are we under-approximating C^- or over-approximating C^+? To answer this question, we performed the following additional experiment. For each of the first 1000 random directions d that yield $\lambda^- < \lambda^+$, we construct a new cost vector

$$c_\pm = c^* + \frac{\lambda^- + \lambda^+}{2} \frac{d}{\|d\|} \in C^+ \setminus C^-,$$

and re-optimize the problem with this new objective. We then count the number of times the original optimal solution (x^*, y^*) remains optimal for the new cost vector $c_\pm$. The next column in Table 1 (OIO, for "original is optimal") reports these counts for all instances except `p0033`, which did not have any directions along which $\lambda^- < \lambda^+$. With the exception of `dsbmip`, `rentacar` and

rgn, the original solution is optimal most of the time, with most instances recording counts above 800 and many getting 1000. These results indicate that the uncertainty region $C^+ \setminus C^-$ is caused primarily by an excessive under-approximation of C^-. They also suggest that (x^*, y^*) is likely to remain optimal inside C^+, even when this fact cannot be guaranteed.

The last experiment also suggests an additional option to explore when (x^*, y^*) is not optimal for the new objective. Algorithm 1 generates a list of solutions that in some sense "surrounds" the original. Therefore, if (x^*, y^*) is not optimal for an objective, one of the solutions from the list may in fact be optimal instead. To test this hypothesis, we designed a final set of experiments. For the first 1000 random directions d with $\lambda^- < \lambda^+$, define $c_\pm$ as before and also define

$$c_+ = c^* + \frac{\lambda^- + 3\lambda^+}{2} \frac{d}{\|d\|} \in \mathbb{R}^n \setminus C^+.$$

Note that just as $c_\pm$ is guaranteed to be in $C^+ \setminus C^-$, c_+ is guaranteed to be in $\mathbb{R}^n \setminus C^+$, and the distance along d from c_+ to C^+ equals the distance from $c_\pm$ to C^-. We re-optimize the problem with both of these new objectives, and count the number of times the best solution from the list is optimal for each objective. In addition, when none of the solutions is optimal, we calculate the relative gap between the best solution in the list and the new optimal value. The final four columns of Table 1 summarize the results, reporting both the number of times the best solution is optimal (BIO) and the average relative difference in objective values (ROD) for non-optimal cases, defined as $\frac{|\text{optimal}-\text{best}|}{|\text{optimal}|}$.

With the exception of bell3a and vpm1, the solution list yields good solutions for the new objective vectors $c_\pm$ and c_+. For c_+, even when none of the solutions is optimal, on average the relative gap is below 2% and significantly below for many instances. Thus, the solutions generated by Algorithm 1 provide an excellent warm start for re-solves, and they can also be used as high-quality heuristic solutions when re-optimization is not possible.

5 Conclusions

We have outlined a procedure that gives polyhedral approximations, with few inequalities, of the stability region of a solution with respect to a set of binary variables with linear objectives. The approximation requires at most a linear number of solves of problems closely related to the original problem $P(c^*)$. Computational experiments with several types of pure and mixed binary programs from MIPLIB 3.0 show that the original optimal solution (x^*, y^*) is likely to remain optimal when the new objective belongs to C^+, even if optimality cannot be guaranteed. In addition, the list of solutions generated as a byproduct of Algorithm 1 can be used effectively to produce a high-quality solution when (x^*, y^*) is not optimal.

Acknowledgement

This research has been supported in part by the National Science Foundation with grant CMMI-0522485.

References

[1] R. E. Bixby, S. Ceria, C. M. McZeal, and M. W. P. Savelsbergh. An Updated Mixed Integer Programming Library: MIPLIB 3.0. Research Report TR98-03, February 1998. Library available on-line at http://miplib.zib.de/miplib3/miplib.html.

[2] C. E. Blair and R. G. Jeroslow. The value function of an integer program. *Mathematical Programming*, 23(1):237–273, 1982.

[3] N. Chakravarti and A. Wagelmans. Calculation of stability radii for combinatorial optimization problems. *Operations Research Letters*, 23(1):1–7, 1998.

[4] M. Hifi, H. Mhalla, and S. Sadfi. Sensitivity of the Optimum to Perturbations of the Profit or Weight of an Item in the Binary Knapsack Problem. *Journal of Combinatorial Optimization*, 10(3):239–260, 2005.

[5] S. Holm and D. Klein. Three methods for postoptimal analysis in integer linear programming. *Mathematical Programming Study*, 21:97–109, 1984.

[6] C.-J. Lin and U.-P. Wen. Sensitivity Analysis of Objective Function Coefficients of the Assignment Problem. *Asia-Pacific Journal of Operations Research*, 24(2):203–221, 2007.

[7] L. Lovász and S. Vempala. Simulated Annealing in Convex Bodies and an $O^*(n^4)$ Volume Algorithm. *Journal of Computer and System Sciences*, 72(2):392–417, 2005.

[8] R. Ramaswamy, J. B. Orlin, and N. Chakravarti. Sensitivity analysis for shortest path problems and maximum capacity path problems in undirected graphs. *Mathematical Programming*, 102(2):355–369, 2005.

[9] L. Schrage and L. Wolsey. Sensitivity Analysis for Branch and Bound Integer Programming. *Operations Research*, 33(5):1008–1023, 1985.

[10] Y. N. Sotskov, V. K. Leontev, and E. N. Gordeev. Some concepts of stability analysis in combinatorial optimization. *Discrete Applied Mathematics*, 58(2):169–190, 1995.

[11] R. E. Tarjan. Sensitivity Analysis of Minimum Spanning Trees and Shortest Path Trees. *Information Processing Letters*, 14(1):30–33, 1982.

[12] S. Van Hoesel and A. Wagelmans. On the complexity of postoptimality analysis of 0/1 programs. *Discrete Applied Mathematics*, 91(1-3):251–263, 1999.

[13] S. Vempala. Geometric Random Walks: A Survey. *MSRI Volume on Combinatorial and Computational Geometry*, 52:573–612, 2005.

Problem	# vars	# bin	# active	$\|c^*\|$	λ^- (avg.)	λ^+ (avg.)	$\frac{\lambda^-(\text{avg.})}{\lambda^+(\text{avg.})}$	$c_\pm \in C^+ \setminus C^-$			$c_+ \in \mathbb{R}^n \setminus C^+$	
								OIO (cnt.)	BIO (cnt.)	ROD (avg.)	BIO (cnt.)	ROD (avg.)
bell3a	133	39	31	1.553E+05	1.129E+05	1.430E+05	0.79	724	724	2.243E-02	0	11.784
bell5	104	30	20	1.609E+05	8947	25875	0.35	1000	1000	-	680	7.174E-04
blend2	353	231	215	45.98	0.08	0.18	0.43	1000	1000	-	1000	-
dcmulti	548	75	71	4220	16.27	17.75	0.92	990	990	8.082E-06	865	1.869E-05
dsbmip	1886	160	14	0	0.48	0.62	0.78	0	1000	-	1000	-
egout	141	55	28	115	3.21	7.76	0.41	1000	1000	-	205	5.296E-03
enigma	100	100	14	1	0.31	0.32	0.95	1000	1000	-	1000	-
fiber	1298	1254	1239	3.167E+06	762	809	0.94	1000	1000	-	990	1.631E-04
fixnet6	878	378	378	5606	1.98	2.28	0.87	1000	1000	-	1000	-
gen	870	144	100	1670	17.51	24.45	0.72	990	1000	-	821	2.544E-05
gesa2_o	1224	384	383	7.623E+06	363	916	0.40	1000	1000	-	1000	-
gesa2	1224	240	240	7.307E+05	1720	2985	0.58	656	656	2.920E-06	64	2.247E-05
gesa3_o	1152	336	336	7.616E+06	1934	70313	0.03	997	997	5.000E-10	0	1.455E-03
gesa3	1152	216	216	6.487E+05	1.104E+04	2.488E+05	0.04	1000	1000	-	0	1.862E-02
gt2	188	24	12	1.302E+04	1180	4906	0.24	1000	1000	-	1000	-
khb05250	1350	24	24	1.225E+07	3.317E+05	6.482E+05	0.51	899	899	2.444E-04	106	1.931E-03
l152lav	1989	1989	1988	8800	3.84	10.04	0.38	1000	1000	-	39	1.216E-04
lseu	89	89	85	1941	4.55	5.22	0.87	1000	1000	-	998	2.667E-04
manna81	3321	18	16	4	0.29	2.44	0.12	1000	1000	-	2	1.139E-04
mas76	151	150	150	1.200E-04	90.91	159.67	0.57	1000	1000	-	961	3.055E-04
misc03	160	159	100	0	101	354	0.29	1000	1000	-	422	1.696E-02
misc06	1808	112	112	0	0.96	1.48	0.65	1000	1000	-	552	8.474E-06
mod008	319	319	311	1316	2.83	7.52	0.38	1000	1000	-	841	1.799E-03
mod010	2655	2655	2433	9350	1.40	4.25	0.33	1000	1000	-	645	7.996E-06
mod011	10958	96	96	1.400E+07	1.211E+05	3.738E+05	0.32	1000	1000	-	121	2.033E-03
modglob	422	98	98	1.292E+05	1753	12852	0.14	1000	1000	-	90	1.079E-04
p0033	33	33	21	945	3.61	3.61	1.00	-	-	-	-	-
p0201	201	201	131	5712	18.02	54.18	0.33	1000	1000	-	1000	-
p0282	282	282	282	2.975E+05	8.92	20.11	0.44	1000	1000	-	435	3.767E-06
p0548	548	548	484	1.420E+04	5.30	15.27	0.35	978	1000	-	1000	-
p2756	2756	2756	2391	3.417E+04	0.35	5.19	0.07	903	1000	-	10	1.219E-04
pk1	86	55	55	0	0.30	0.43	0.69	1000	1000	-	1000	-
pp08a	240	64	64	2437	11.18	15.59	0.72	1000	1000	-	485	4.077E-04
pp08aCUTS	240	64	64	2437	11.18	15.59	0.72	1000	1000	-	486	3.969E-04
qnet1_o	1541	1288	1260	0	0	8.03	0.00	1000	1000	-	996	2.697E-05
qnet1	1541	1288	1260	0	7.79	8.03	0.97	1000	1000	-	910	2.760E-05
rentacar	9557	55	22	0	6.020E+04	6.098E+04	0.99	0	1000	-	437	1.314E-04
rgn	180	100	60	0	5.67	7.93	0.71	0	1000	-	1000	-
set1ch	712	240	235	1.094E+04	8.52	11.73	0.73	1000	1000	-	854	1.509E-05
vpm1	378	168	114	10.68	0.11	4.33	0.03	960	960	4.330E-03	1	0.236
vpm2	378	168	136	10.21	0.08	0.56	0.14	834	1000	-	838	3.625E-03

Table 1: Shooting experiment results for MIPLIB 3.0 instances.