Approximating the Stability Region for Binary Mixed-Integer Programs

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

School of Industrial and Systems Engineering, Georgia Institute of Technology, 765 Ferst Drive NW, Atlanta, GA, 30032

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 of such a problem is the polyhedral set of objective coefficients for which the solution is optimal. A priori knowledge of this set provides valuable information for sensitivity analysis and re-optimization when there is objective coefficient uncertainty. An exact description of the stability region typically requires an exponential number of inequalities. We develop useful polyhedral inner and outer approximations of the stability region using only a linear number of inequalities. Furthermore, when a new objective function is not in the stability region, we produce a list of good solutions that can be used as a quick heuristic or as a warm start for future solves.

Key words: Real-time optimization, stability analysis, binary integer programming.

1 Introduction

As computational and theoretical advances continue, it has become increasingly possible and desirable to perform sensitivity and stability analysis on NP-hard optimization problems in a manner that would have been impractical even a few years ago. In this context, our investigation is motivated by the following scenario. Suppose we have an optimization problem with some

Email addresses: `fkilinc@isye.gatech.edu` (Fatma Kılınç-Karzan), `atoriell@isye.gatech.edu` (Alejandro Toriello), `sahmed@isye.gatech.edu` (Shabbir Ahmed), `gnemhaus@isye.gatech.edu` (George Nemhauser), `mwps@isye.gatech.edu` (Martin Savelsbergh).

13 November 2007

binary variables, say $x \in \{0,1\}^n$, that contribute linearly to the problem's objective. We are given an initial cost vector c^*, which estimates the x variables' contribution to the objective. However, the x variables' true cost deviates within some predetermined ball around c^*. Furthermore, we suppose that we have ample time in which to solve the problem with estimated cost vector c^* and extract any other information deemed useful or desirable. However, once the true cost vector c becomes known, we must quickly, possibly in real time, verify if the solution to the problem with estimated cost c^* is still optimal, and, if not, produce a solution to the problem with true cost c. In the latter situation, we can make use of any information extracted during the initial optimization to expedite re-optimization. Note that our situation varies from other existing frameworks developed to deal with uncertainty, such as robust or stochastic optimization. Robust and stochastic optimization seek solutions with good objective values for a large family of problem instances, while we want to optimize a specific problem instance, in real time.

In our model, where only the objective coefficients of the x variables are uncertain, the stability region of a solution is the set of objective coefficients in n-space for which the solution is optimal. It is easy to see that the stability region is a polyhedron. However, this polyhedron may be defined by a number of inequalities that is proportional to the number of feasible solutions (with respect to the x variables) and therefore may be exponential in n.

Our main contribution is an approximation of the stability region by inner and outer polyhedra each described by a number of inequalities proportional to n, so that it is easy to test whether cost vectors are in these polyhedra. Points in the inner polyhedron correspond to objectives for which the current solution is optimal and points not in the outer polyhedron correspond to objectives for which the current solution is not optimal. However, the approximations give no information on points between the two polyhedra, which we call the uncertainty region. We give computational results on MIPLIB problems that demonstrate the effectiveness of the approximations. As an additional contribution, if the original solution is not optimal, the algorithm that produces the approximation also yields feasible solutions that can be used as a quick heuristic or as a warm start for future solves.

Our analysis can accommodate a very general optimization problem, only requiring that the variables under scrutiny be binary and that their contribution to the problem's objective be linear. Other researchers have, on the other hand, studied specific combinatorial optimization problems. Shortest paths and maximum capacity paths are studied in [14], assignment problems in [12], minimum spanning trees and shortest path trees in [17], and knapsacks in [9]. In addition, right-hand side vector sensitivity in integer and mixed-integer linear programs is studied in [5,10,15]. Related complexity issues are studied in [6,8,13,18]. A survey of results in stability analysis of combinatorial optimization problems appears in [16]. An algorithm for parametric objective function analysis of binary integer linear programs is given in [7]. Finally, [11] considers a problem setting similar to ours, but the results do not focus on the stability region and relate to our results only in the production of good solutions for the new objective.

The rest of the paper is organized as follows. Section 2 introduces the concepts and terminology we use along with basic results. Section 3 presents our approximations of the stability region. Section 4 places the approximation results in an algorithmic framework. Section 5 gives computational results. Section 6 has closing remarks and some ideas for future research.

2 The Stability Region

Consider the optimization problem

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

where $c^*x = \sum_{i\in N} c^*_i x_i$ and $N = \{1, \ldots, n\}$. By possibly complementing the x variables, we assume without loss of generality that (x^*, y^*) is an optimal solution with $x^* = 0$ and objective value $z^* = c^*x^* + h(y^*) = h(y^*)$. We are interested in the stability of (x^*, y^*) with respect to perturbations of c^*.

Throughout this paper, when we refer to problem $P(c^*)$, we mean the optimization problem with cost vector c^* for the x variables. When we refer to problem $P(c)$, for some $c \in \mathbb{R}^n$, we mean the optimization problem with the same feasible region, same objective function $h(y)$ for the y variables, but cost vector c for the x variables.

The following observation will be useful in subsequent analysis.

Lemma 2.1 *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 (x^*, y^*) is not optimal for $P(\hat{c})$.

Proof. For $P(\hat{c})$, the objective function value of $(\hat{x}, \hat{y})$ is $\hat{c}\hat{x} + h(\hat{y}) = \hat{z} + (\hat{c} - c^*)\hat{x} > z^* = \hat{c}x^* + h(y^*)$. ■

Definition 2.1 *Consider the optimization problem $P(c^*)$ with optimal solu-*

tion (x^*, y^*). *The* stability region *of* (x^*, y^*) *is the region* $C \subseteq \mathbb{R}^n$ *s.t.* $c \in C$ *iff* (x^*, y^*) *is optimal for* $P(c)$. *That is,*

$$C = \{c \in \mathbb{R}^n : cx \leq h(y^*) - h(y), \forall\ (x, y) \in X\}.$$

Lemma 2.2 *Let* $\hat{x} \in \{0,1\}^n$, *and define the optimization problem*

$$\begin{aligned} \upsilon(\hat{x}) = \max\ & h(y) \\ \text{s.t. } & (\hat{x}, y) \in X. \end{aligned}$$

If $\hat{x} \notin \text{proj}_x(X) = \{x \in \{0,1\}^n : \exists\ y \text{ s.t. } (x,y) \in X\}$, *define* $\upsilon(\hat{x}) = -\infty$. *Then*

$$C = \{c \in \mathbb{R}^n : cx \leq h(y^*) - \upsilon(x), \forall\ x \in \{0,1\}^n\}.$$

Proof. Let $(\hat{x}, \hat{y}) \in X$. The proof follows directly from the fact that $\upsilon(\hat{x}) \geq h(\hat{y})$. ■

Example 1 *Consider the binary IP*

$$\begin{aligned} \max\ & 4y_1 + 2y_2 + y_3 \\ \text{s.t. } & x_1 + y_1 \leq 1 \\ & x_2 + y_2 \leq 1 \\ & -x_1 + y_3 \leq 0 \\ & -x_2 + y_3 \leq 0 \\ & x_1, x_2, y_1, y_2, y_3 \in \{0,1\}. \end{aligned} \tag{1}$$

The optimal solution is $(x^*, y^*) = (0, 0, 1, 1, 0)$ *with* $z^* = 6$. *Similarly,*

$$\upsilon((1,0)) = 2, \quad \upsilon((0,1)) = 4, \quad \upsilon((1,1)) = 1.$$

The stability region C *is defined by the three inequalities*

$$c_1 \leq 6 - 2 = 4, \quad c_2 \leq 6 - 4 = 2, \quad c_1 + c_2 \leq 6 - 1 = 5.$$

Figure 1 illustrates C.

Lemma 2.2 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

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

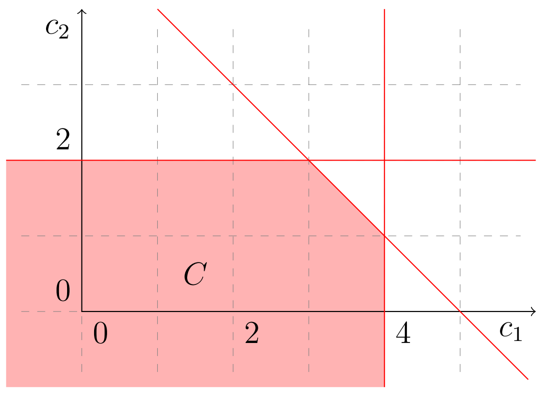


Fig. 1. Stability region for (x^*, y^*) in problem (1).

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 [6,16]. 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 2.2 *An* outer optimality approximation neighborhood *of* (x^*, y^*) *is a region* $C^+ \subseteq \mathbb{R}^n$ *satisfying* $C^+ \supseteq C$*, where* C *is the stability region of* (x^*, y^*)*. For simplicity, we refer to* C^+ *as an outer neighborhood.*

For any outer neighborhood C^+, if $c \notin C^+$, then (x^*, y^*) is not optimal for $P(c)$. From Lemma 2.1, the region $\{c \in \mathbb{R}^n : c\hat{x} \leq z^* - \hat{z} + c^*\hat{x}\}$ is an outer neighborhood of (x^*, y^*).

Definition 2.3 *An* inner optimality approximation neighborhood *of* (x^*, y^*) *is a region* $C^- \subseteq \mathbb{R}^n$ *satisfying* $C^- \subseteq C$*. We refer to* C^- *as an inner neighborhood.*

For any inner neighborhood C^-, if $c \in C^-$, then (x^*, y^*) is optimal for $P(c)$. Trivially, the region $\{c \in \mathbb{R}^n : c \leq c^*\}$ is an inner neighborhood of (x^*, y^*), once again assuming $x^* = 0$.

The following are two simple but important consequences of Definitions 2.2 and 2.3 that help us obtain small outer neighborhoods and large inner neighborhoods.

Proposition 2.3

i) If C_1^+, C_2^+ *are outer neighborhoods,* $C_1^+ \cap C_2^+$ *is an outer neighborhood.*
ii) If C_1^-, C_2^- *are inner neighborhoods,* $\operatorname{conv}(C_1^- \cup C_2^-)$ *is an inner neighborhood.*

Proof. Part (*i*) is an immediate consequence of our definitions. For (*ii*), let $(\hat{x}, \hat{y}) \in X, c^1 \in C_1^-, c^2 \in C_2^-$, and $\lambda \in [0,1]$. By definition we have

$$c^j x^* + h(y^*) \geq c^j \hat{x} + h(\hat{y}), j = 1, 2.$$

Multiplying the first inequality by λ, the second by $(1-\lambda)$, and adding them yields $\lambda c^1 + (1-\lambda)c^2 \in C$. ■

3 Approximating the Stability Region

In this section, we will show how to obtain inner and outer neighborhoods of (x^*, y^*) by solving the n problems $\{P_j : j \in N\}$, where each P_j is given by

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

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 $P_j, \forall\, j \in N$ and determined optimal solutions (x^j, y^j) with corresponding objective values z_j.

The following observation follows directly from Lemma 2.1 and Proposition 2.3.

Proposition 3.1 *The set*

$$C^+ = \left\{c \in \mathbb{R}^n : cx^j \leq z^* - z_j + c^* x^j, \forall\, j \in N\right\}$$

is an outer neighborhood of $P(c^*)$.

Let $\gamma_j = z^* - z_j + c_j^*$. Observe that the outer neighborhood C^+ satisfies

$$\{c \in C^+ : c \geq c^*\} \subseteq \{c \in \mathbb{R}^n : c_j^* \leq c_j \leq \gamma_j, \forall\, j \in N\}.$$

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

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

Proposition 3.2 *Let*

$$\tilde{c}^j = (c_1^*, \ldots, c_{j-1}^*, \gamma_j, c_{j+1}^*, \ldots, c_n^*).$$

The solution (x^*, y^*) is optimal for $P(\tilde{c}^j)$, $\forall\ j \in N$.

Proof. To begin, 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

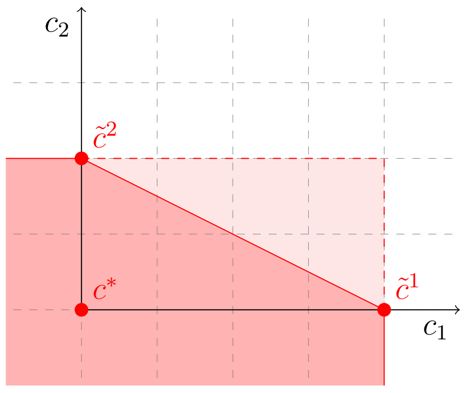
$$\tilde{c}^j \hat{x} + h(\hat{y}) = c^* \hat{x} + h(\hat{y}) = \hat{z} \leq z^*,$$

by assumption. If $\hat{x}_j = 1$, then

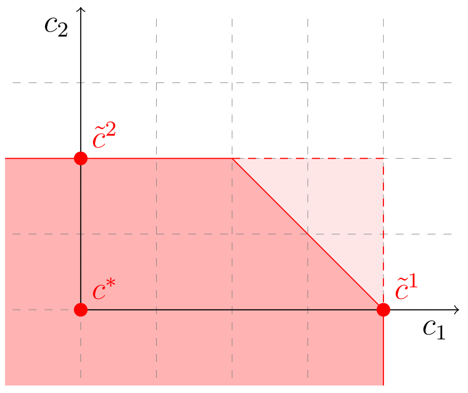
$$\tilde{c}^j \hat{x} + h(\hat{y}) = c^* \hat{x} + h(\hat{y}) - z_j + z^* = \underbrace{\hat{z} - z_j}_{\leq 0} + z^* \leq z^*,$$

where the last inequality follows because $(\hat{x}, \hat{y})$ is feasible for P_j. ■

We now have a list of cost vectors $\{\tilde{c}^j\}_{j \in N}$ for which we have proved the optimality of (x^*, y^*). These points together with c^* form the vertices of a simplex, which by Proposition 2.3 is an inner neighborhood of (x^*, y^*) (see Figure 2a.) However, from Lemma 2.2, we also know that every inequality in the description of the stability region is of the form $cx \leq h(y^*) - \upsilon(x)$, for some $x \in \text{proj}_x(X)$. We can therefore exploit the known structure of these inequalities to increase the inner neighborhood, as in Figure 2b. The following theorem formalizes this notion.



(a)



(b)

Fig. 2. The convex hull of c^* and $\{\tilde{c}^j\}_{j \in N}$ yields an inner neighborhood, as in diagram (a). We can use the known structure of the stability region's cuts to expand the neighborhood, as seen in (b). Both regions can be expanded by adding the negative orthant to every point, as explained in Corollary 3.4.

Theorem 3.3 *Suppose we order the x variables so that*

$$z^* \geq z_1 \geq z_2 \geq \cdots \geq z_n$$

holds. Then

$$C^{-} = \left\{ c \geq c^* : \sum_{i=1}^{j} c_i \leq \gamma_j + \sum_{i=1}^{j-1} c_i^*, \forall\ j \in N \right\}$$

is an inner neighborhood of (x^*, y^*)*.*

Proof. For any $x \in \{0,1\}^n$, we do not know the exact value of the right-hand side in the inequality $cx \leq h(y^*) - v(x)$. However, Proposition 3.2 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

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

is an inner neighborhood of $P(c^*)$. 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 1 *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

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

■

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

$$\sum_{i\in I} c_i \le z^* - z_t + \sum_{i\in I} c_i^* + \underbrace{\sum_{i\in\{1,\dots,t\}\setminus I} (c_i^* - c_i)}_{\le 0}$$
$$\le z^* - z_t + \sum_{i\in I} c_i^* = \max\left\{\sum_{i\in I} \tilde{c}_i^j : j \in N\right\}.$$

■

Corollary 3.4 *The set $\{c + d : c \in C^-, d \le 0\}$ is an inner neighborhood of $P(c^*)$.*

Proof. After Definition 2.3, we noted that if (x^*, y^*) (with $x^* = 0$) is optimal for $P(c^*)$, then the set $\{c \in \mathbb{R}^n : c \le c^*\}$ is a trivial inner neighborhood of $P(c^*)$. Using Proposition 2.3, we apply this observation to C^-. ■

To test membership of any vector $c \in \mathbb{R}^n$ in this extended set, it suffices to replace c_i with $\max\{c_i, c_i^*\}, \forall\, i \in N$ and test the membership of the resulting vector in C^-.

These last two results motivate a natural algorithm for determining an inner neighborhood. Solve each of the problems P_j in turn, sort them by objective value, and compute the inner neighborhood as indicated in Theorem 3.3. As we shall see in the next section, this procedure can be modified slightly to potentially reduce the number of solves while still yielding the same region.

4 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 $\sum_{i\in N} x_i \ge 1$, 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.) For future reference, we formally define the relevant information gathered during the execution of Algorithm 1. 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^+ indicates which binary variables have value one in (x^k, y^k). The set I_k^- indicates which of these variables have value one for the first time.

An outer neighborhood is easily obtained applying Lemma 2.1 to each solution (x^k, y^k). To determine an inner neighborhood, we must first establish a preliminary fact.

Algorithm 1 Binary Solution Cover

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

  Set $(x^0, y^0) \leftarrow (x^*, y^*)$, $z_0 \leftarrow z^*$, $I \leftarrow N$.
  Add cut $D \equiv (\sum_{i \in I} x_i \geq 1)$ to $P(c^*)$.
  for $k = 1, \dots, n$ 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
```

Proposition 4.1 *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 4.1 with Theorem 3.3 to construct our inner neighborhood.

Theorem 4.2 *Suppose Algorithm 1 is run on problem $P(c^*)$, terminating after $\ell \leq n$ steps. Let $(x^k, y^k), z_k, I_k^+, I_k^-, \forall\ k = 1, \dots, \ell$ and I_∞ be obtained from the algorithm. Then*

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

is an inner neighborhood of $P(c^*)$.

Proof. As in Section 3, we assume without loss of generality that $I_\infty = \emptyset$. (That is, we assume $\exists$ a feasible solution to $P(c^*)$ with $x_j = 1, \forall\ j \in N$.) To begin, notice that the sets $I_k^-, \forall\ k = 1, \dots, \ell$ are pairwise disjoint and form a partition of N. By possibly rearranging indices, we may also assume that

$$\begin{aligned} I_1^- &= \{1, \ldots, t_1\} \\ I_1^- \cup I_2^- &= \{1, \ldots, t_2\} \\ &\vdots \\ I_1^- \cup \cdots \cup I_\ell^- &= \{1, \ldots, n\} = N, \end{aligned}$$

for properly chosen indices $t_1 < \cdots < t_\ell = n$.

The proof now follows directly from Theorem 3.3. Let $I \subseteq N$ and let $k \in \{1, \ldots, \ell\}$ be the smallest iteration index satisfying $I \subseteq I_1^- \cup \cdots \cup I_k^-$. We only need to prove that

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

So let $s \in N$. If $s \leq t_{k-1}$ or $s > t_k$, the proof is exactly as before. The only new case is when $t_{k-1} < s \leq t_k$. In other words, this means $s \in I_k^-$; but then $(x^s, y^s) = (x^{t_k}, y^{t_k}) = (x^k, y^k)$, yielding

$$\sum_{i \in I} \tilde{c}_i^s = \sum_{i \in I} \tilde{c}_i^{t_k},$$

which finishes the proof. ■

We can also apply Corollary 3.4 to get an inner neighborhood 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 neighborhood we calculate using Algorithm 1 does depend on c^*, since 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 vector for which (x^*, y^*) is optimal can be used in the algorithm to obtain a possibly different inner neighborhood. Moreover, even if the solutions and their respective orders in the algorithm remain the same, the right hand side of the inequalities can be different for different c vectors. In other words, for any $k = 1, \ldots, \ell$,

$$\begin{aligned} \sum_{i \in I_1^- \cup \cdots \cup I_k^-} c_i &\leq z^* - z_k + \sum_{i \in I_1^- \cup \cdots \cup I_k^-} c_i^* \\ &= h(y^*) - [c^* x^k + h(y^k)] + \sum_{i \in I_1^- \cup \cdots \cup I_k^-} c_i^* \\ &= h(y^*) - h(y^k) - \sum_{i \in I_k^+} c_i^* + \sum_{i \in I_1^- \cup \cdots \cup I_k^-} c_i^*. \end{aligned}$$

By the construction of I_k^- and I_k^+, we have $I_k^+ \subseteq \left(I_1^- \cup \cdots \cup I_k^-\right)$, so we obtain

$$\sum_{i \in I_1^- \cup \cdots \cup I_k^-} c_i \leq h(y^*) - h(y^k) + \sum_{i \in \left(I_1^- \cup \cdots \cup I_k^-\right) \setminus I_k^+} c_i^* .$$

Clearly $\left(I_1^- \cup \cdots \cup I_k^-\right) \setminus I_k^+$ can be nonempty depending on the solutions obtained and their order. This establishes that two inner neighborhoods obtained with different cost vectors may be different even when the algorithm returns the same solutions in the same order for both vectors.

Example 2 *Consider the (unconstrained) binary linear program*

$$\begin{aligned} \max\ & c_1 x_1 + c_2 x_2 \\ s.t.\ & x_1, x_2 \in \{0,1\}. \end{aligned} \tag{2a}$$

Clearly, $x^* = (0,0)$ is optimal iff $c \leq 0$. If we use $c^* = (0,0)$ as the starting cost in Algorithm 1 and apply Corollary 3.4, we will obtain precisely the negative quadrant. However, if the starting cost is $c^* = (-1,-1)$, Algorithm 1 and Corollary 3.4 yield the truncated region $C^- = \{c \leq 0 : c_1 + c_2 \leq -1\}$ (see Figure 3a.)

Now consider another binary linear program

$$\begin{aligned} \max\ & c_1 x_1 + c_2 x_2 \\ s.t.\ & x_1 = x_2 \\ & x_1, x_2 \in \{0,1\}. \end{aligned} \tag{2b}$$

This time, x^* is optimal iff $c_1 + c_2 \leq 0$. If $c^* = (0,0)$ is the starting cost, we will once again obtain only the negative quadrant. On the other hand, the starting cost $c^* = (-1,-1)$ returns the region $C^- = \{c \in \mathbb{R}^2 : c_1 + c_2 \leq 0, c_1 \leq 1, c_2 \leq 1\}$, which contains the negative quadrant (see Figure 3b.)

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 4.3 essentially states that applying the algorithm to the larger set N does not yield better results.

Proposition 4.3 *Suppose a run of Algorithm 1 on N produces the solution list $\{(x^k, y^k)\}_{k=1}^{\ell}$ 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 neighborhood C_M^- satisfies*

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

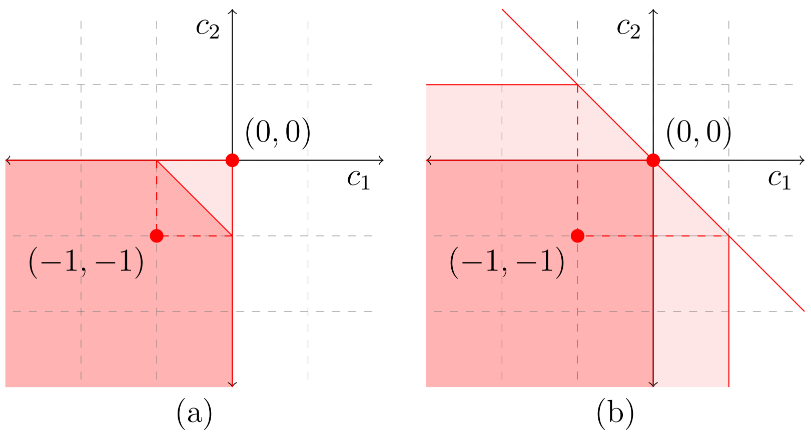


Fig. 3. The inner neighborhood returned by Algorithm 1 and Corollary 3.4 varies depending on starting cost.

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

$$C_N^- \cap \{c : c_i = c_i^*, \forall\, i \in N \setminus M\} =$$
$$\left\{ c \geq c^* : \sum_{i \in \bigcup_{j=1}^k J_j} c_i \leq z^* - z_k + \sum_{i \in \bigcup_{j=1}^k J_j} c_i^*, \forall\, k; c_i = c_i^*, \forall\, i \in N \setminus M \right\} =$$
$$\left\{ c \geq c^* : \sum_{i \in \bigcup_{j=1}^k J_j} c_i \leq z^* - z_k + \sum_{i \in \bigcup_{j=1}^k J_j} c_i^*, \forall\, J_k^- \neq \emptyset; c_i = c_i^*, \forall\, i \in N \setminus M \right\},$$

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 4.3 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.

5 Computational Results

In this section, we provide and interpret the results of a set of experiments designed to evaluate Algorithm 1. The goal is to study the quality of the inner and outer approximations of the stability region, and to test the value of the solutions obtained during the execution of the algorithm when re-optimization is necessary.

The set of instances used in our computational experiments contains pure and mixed-integer linear programs from MIBLIB 3.0 [4]. All computational experiments were carried out on a system with two 2.4 GHz Xeon processors and 2 GB RAM, and using CPLEX 9.0 as the optimization engine.

For each instance, we take all binaries as x variables, i.e., variables under scrutiny, and all others as y variables. We first solve the original instance to find an optimal solution, (x^*, y^*), and then we use Algorithm 1 to compute (x^k, y^k), z_k, I_k^+, and I_k^-, which in turn can be used to derive an inner (C^-) and outer (C^+) approximation of C. The instance characteristics and a summary of computations can be found in Table 1, where we report the number of variables (binary, general integer, continuous) in each instance, the number of solutions generated during the execution of Algorithm 1, and the number of variables which are fixed to their original value ($|I_\infty|$).

Recall that for $i \in I_\infty$, the corresponding objective function coefficient c_i can vary arbitrarily in either direction without changing the optimality of (x^*, y^*). Therefore, in our analysis of the quality of the inner and outer approximation of C, we restrict cost vector perturbations to binary variables x_i with $i \notin I_\infty$ (i.e., we fix the costs of variables in I_∞ to c_i^*). We also know that if $x_i^* = 0$ and we decrease c_i^*, then (x^*, y^*) will remain optimal. Therefore, we randomly perturb each c_i within $p\%$ of its original value c_i^* in the direction of interest, using independent uniform random variables. That is, since $x_i^* = 0$, we set $\hat{c}_i = c_i^*(1 + \text{sign}(c_i^*)0.01pU_i)$, where $U_i \backsim U(0, 1)$. For each instance and for $p = 1, 2, 5, 10, 20$, we examine 1000 perturbed cost vectors.

The experimental design influences problem selection in various ways. To begin, we obviously need $N \setminus I_\infty \neq \emptyset$; that is, we need at least one binary variable that is not fixed to its initial value. Furthermore, of the variables in the set $N \setminus I_\infty$, at least one must have a non-zero cost, since our perturbations are proportional to initial costs. In fact, we ensure that at least one variable from $N \setminus I_\infty$ has positive cost, because if all costs are negative, our perturbations will never leave the negative orthant, which would bias the results for this instance.

Note that the perturbed cost vector $\hat{c}$ can be in C^-, $C^+ \setminus C^-$, or $\mathbb{R}^n \setminus C^+$. If $\hat{c} \in C^-$, then (x^*, y^*) remains optimal for the new objective function. If

Problem	Binary	General Integer	Continuous	Solutions	$\lvert I_\infty \rvert$
dcmulti	75	0	473	60	3
egout	55	0	86	13	27
gen	144	6	720	83	36
khb05250	24	0	1326	18	0
lseu	89	0	0	65	0
mod008	319	0	0	309	0
modglob	98	0	324	81	0
p0033	33	0	0	12	4
p0201	201	0	0	73	6
p0282	282	0	0	246	0
pp08a	64	0	176	40	0
qiu	48	0	792	10	0
stein27	27	0	0	7	0
vpm2	168	0	210	139	2

Table 1
Problem set information.

$\hat{c} \in \mathbb{R}^n \setminus C^+$, then the optimal solution will change. Finally, in the region $C^+ \setminus C^-$, the optimality of (x^*, y^*) is undetermined. Whenever $\hat{c} \in \mathbb{R}^n \setminus C^-$, we need to reoptimize. We investigate what happens when instead of actually reoptimizing, we simply settle for the best solution (with respect to the perturbed cost vector) among the solutions generated during the execution of Algorithm 1.

Tables 2 through 11 provide the results of our computational experiments. We report the following statistics:

- The number of times a perturbed cost vector $\hat{c}$ falls into the regions C^-, $C^+ \setminus C^-$ or $\mathbb{R}^n \setminus C^+$.
- The number of times the best solution among $\{(x^0, y^0), \ldots, (x^\ell, y^\ell)\}$, i.e., the solutions generated by Algorithm 1, is optimal for the perturbed cost vector $\hat{c}$.

If for at least one perturbed cost vector $\hat{c}$ neither the original solution nor the solutions produced by Algorithm 1 are optimal, we also report the following statistics:

- The mean and standard deviation of $\frac{(\text{optimal}-\text{best})}{|\text{optimal}|}$, the relative error in objective value, calculated over instances where $\hat{c} \in \mathbb{R}^n \setminus C^-$.

	Region Counts			Times Best is Optimal		
Problem	C^-	$C^+ \setminus C^-$	$\mathbb{R}^n \setminus C^+$	$C^+ \setminus C^-$	$\mathbb{R}^n \setminus C^+$	Total
`dcmulti`	1000	0	0	-	-	1000
`egout`	1000	0	0	-	-	1000
`gen`	0	0	1000	-	993	993
`khb05250`	1000	0	0	-	-	1000
`lseu`	0	0	1000	-	855	855
`mod008`	0	0	1000	-	485	485
`modglob`	1000	0	0	-	-	1000
`p0033`	0	0	1000	-	278	278
`p0201`	0	0	1000	-	1000	1000
`p0282`	1000	0	0	-	-	1000
`pp08a`	281	669	50	634	25	940
`qiu`	0	0	1000	-	39	39
`stein27`	0	0	1000	-	0	0
`vpm2`	0	0	1000	-	315	315

Table 2
Region and optimality counts for 1% perturbations.

	$\frac{(\text{optimal}-\text{best})}{\lvert\text{optimal}\rvert}$		95% CI on Relative Diff.	
Problem	Average	St. Dev.	LB	UB
`gen`	3.29E-08	5.03E-07	3.19E-08	3.39E-08
`lseu`	3.99E-05	1.28E-04	3.96E-05	4.01E-05
`mod008`	5.85E-04	8.18E-04	5.83E-04	5.86E-04
`p0033`	3.97E-04	4.01E-04	3.96E-04	3.98E-04
`pp08a`	9.84E-06	4.28E-05	9.76E-06	9.93E-06
`qiu`	8.02E-03	4.50E-03	8.01E-03	8.03E-03
`stein27`	1.33E-03	3.76E-04	1.33E-03	1.34E-03
`vpm2`	2.75E-04	3.23E-04	2.74E-04	2.75E-04

Table 3
Relative difference results for 1% perturbations, where relevant.

- The 95% confidence interval on the relative error in objective value over instances where $\hat{c} \in \mathbb{R}^n \setminus C^-$.

	Region Counts			Times Best is Optimal		
Problem	C^-	$C^+ \setminus C^-$	$\mathbb{R}^n \setminus C^+$	$C^+ \setminus C^-$	$\mathbb{R}^n \setminus C^+$	Total
`dcmulti`	1000	0	0	-	-	1000
`egout`	1000	0	0	-	-	1000
`gen`	0	0	1000	-	993	993
`khb05250`	1000	0	0	-	-	1000
`lseu`	0	0	1000	-	764	764
`mod008`	0	0	1000	-	485	485
`modglob`	1000	0	0	-	-	1000
`p0033`	0	0	1000	-	49	49
`p0201`	0	0	1000	-	1000	1000
`p0282`	1000	0	0	-	-	1000
`pp08a`	0	4	996	2	356	358
`qiu`	0	0	1000	-	39	39
`stein27`	0	0	1000	-	0	0
`vpm2`	0	0	1000	-	315	315

Table 4
Region and optimality counts for 2% perturbations.

	$\frac{(\text{optimal}-\text{best})}{\lvert\text{optimal}\rvert}$		95% CI on Relative Diff.	
Problem	Average	St. Dev.	LB	UB
`gen`	3.29E-08	5.03E-07	3.19E-08	3.39E-08
`lseu`	1.26E-04	3.13E-04	1.25E-04	1.27E-04
`mod008`	5.85E-04	8.18E-04	5.83E-04	5.86E-04
`p0033`	1.67E-03	8.73E-04	1.66E-03	1.67E-03
`pp08a`	2.11E-04	2.46E-04	2.11E-04	2.12E-04
`qiu`	1.55E-02	8.65E-03	1.54E-02	1.55E-02
`stein27`	2.67E-03	7.51E-04	2.67E-03	2.67E-03
`vpm2`	2.75E-04	3.23E-04	2.74E-04	2.75E-04

Table 5
Relative difference results for 2% perturbations, where relevant.

	Region Counts			Times Best is Optimal		
Problem	C^-	$C^+ \setminus C^-$	$\mathbb{R}^n \setminus C^+$	$C^+ \setminus C^-$	$\mathbb{R}^n \setminus C^+$	Total
`dcmulti`	1000	0	0	-	-	1000
`egout`	783	217	0	217	-	1000
`gen`	0	0	1000	-	631	631
`khb05250`	360	640	0	627	-	987
`lseu`	0	0	1000	-	760	760
`mod008`	0	0	1000	-	80	80
`modglob`	0	1000	0	1000	-	1000
`p0033`	0	0	1000	-	8	8
`p0201`	0	0	1000	-	1000	1000
`p0282`	50	950	0	950	-	1000
`pp08a`	0	0	1000	-	393	393
`qiu`	0	0	1000	-	37	37
`stein27`	0	0	1000	-	0	0
`vpm2`	0	0	1000	-	315	315

Table 6
Region and optimality counts for 5% perturbations.

	$\frac{(\text{optimal}-\text{best})}{\lvert\text{optimal}\rvert}$		95% CI on Relative Diff.	
Problem	Average	St. Dev.	LB	UB
`gen`	1.06E-05	1.78E-05	1.05E-05	1.06E-05
`khb05250`	4.46E-06	3.81E-05	4.38E-06	4.53E-06
`lseu`	3.25E-04	8.06E-04	3.23E-04	3.26E-04
`mod008`	5.91E-03	3.89E-03	5.90E-03	5.91E-03
`p0033`	5.12E-03	1.96E-03	5.12E-03	5.12E-03
`pp08a`	3.03E-04	3.96E-04	3.02E-04	3.04E-04
`qiu`	3.49E-02	1.94E-02	3.49E-02	3.50E-02
`stein27`	6.68E-03	1.88E-03	6.67E-03	6.68E-03
`vpm2`	6.87E-04	8.07E-04	6.85E-04	6.89E-04

Table 7
Relative difference results for 5% perturbations, where relevant.

	Region Counts			Times Best is Optimal		
Problem	C^-	$C^+ \setminus C^-$	$\mathbb{R}^n \setminus C^+$	$C^+ \setminus C^-$	$\mathbb{R}^n \setminus C^+$	Total
`dcmulti`	509	313	178	309	71	889
`egout`	0	814	186	814	186	1000
`gen`	0	0	1000	-	432	432
`khb05250`	0	344	656	227	165	392
`lseu`	0	0	1000	-	712	712
`mod008`	0	0	1000	-	26	26
`modglob`	0	1000	0	1000	-	1000
`p0033`	0	0	1000	-	4	4
`p0201`	0	0	1000	-	973	973
`p0282`	0	509	491	509	491	1000
`pp08a`	0	0	1000	-	474	474
`qiu`	0	0	1000	-	32	32
`stein27`	0	0	1000	-	0	0
`vpm2`	0	0	1000	-	269	269

Table 8
Region and optimality counts for 10% perturbations.

Note that for 1% and 2% perturbations, in every instance except `pp08a` we encounter no uncertainty. For various instances, such as `dcmulti`, `egout` and `p0201`, the solution list provided by Algorithm 1 contains an optimal solution for every single random cost vector. Moreover, for these two perturbation levels, our relative difference averages are all under 1%, except for `qiu` at the 2% perturbation level, which has a 1.55% relative difference average.

For higher perturbation levels (5%, 10% and 20%) uncertainty counts increase and optimality counts decrease, but the solution list consistently contains solutions with a value that is close to the optimal value. For example, at perturbation levels 10% and 20%, only problem `dcmulti` still has random cost vectors in the inner approximation C^-, but the solution list provides an optimal solution for at least 500 of the 1000 random cost vectors in several instances, such as `lseu`, `modglob` and `p0282`. Also, even at the 20% perturbation level, all instances but one (`qiu`) have average relative difference below 10%, and many (`gen`, `modglob`, `p0201`, etc.) have this average still below 0.1%.

The experimental results show that, for smaller perturbations, the probability that $\hat{c} \in C^+ \setminus C^-$ is relatively low for most instances. Furthermore, the average and standard deviations of the relative error between the optimal solution and

	$\frac{(\text{optimal}-\text{best})}{\lvert\text{optimal}\rvert}$		95% CI on Relative Diff.	
Problem	Average	St. Dev.	LB	UB
`dcmulti`	1.76E-06	3.29E-06	1.75E-06	1.76E-06
`gen`	4.98E-05	6.07E-05	4.97E-05	4.99E-05
`khb05250`	5.35E-04	6.58E-04	5.34E-04	5.37E-04
`lseu`	8.00E-04	1.76E-03	7.97E-04	8.04E-04
`mod008`	1.85E-02	9.76E-03	1.85E-02	1.85E-02
`p0033`	1.07E-02	3.81E-03	1.07E-02	1.07E-02
`p0201`	3.89E-05	3.17E-04	3.83E-05	3.95E-05
`pp08a`	4.11E-04	6.07E-04	4.10E-04	4.12E-04
`qiu`	6.12E-02	3.30E-02	6.11E-02	6.12E-02
`stein27`	1.34E-02	3.76E-03	1.33E-02	1.34E-02
`vpm2`	1.61E-03	1.75E-03	1.61E-03	1.62E-03

Table 9
Relative difference results for 10% perturbations, where relevant.

the best generated solution is small for most instances, and for many the best solution is in fact optimal. This suggests that simply using the best solution among the solutions generated during the execution of Algorithm 1 provides an excellent alternative to actual re-optimization when it is too costly or too time consuming to do so.

5.1 Comparing the Approximations to the Stability Region

In this subsection, we further test the quality of the approximations generated by Algorithm 1 by comparing C^- and C^+ to the exact stability region C. Although we cannot efficiently compute the exact stability region of (x^*, y^*) when $P(c^*)$ is NP-hard, if the convex hull of the feasible region is given by polynomially many inequalities, as is the case when the constraint matrix is totally unimodular and the formulation has polynomial size, we can use well-known linear programming theory to generate the region. Specifically, suppose

$$x^* \in S = \{x \in \mathbb{R}^n : a^i x \leq b_i, \forall\ i = 1, \ldots, m\},$$

where $a^i \in \mathbb{R}^n$ and $b_i \in \mathbb{R}$. Then by complementary slackness and LP duality, x^* is optimal for $\max\{cx : x \in S\}$ iff $c \in C = \text{cone}\{a_i : i \in I^*\}$, where $I^* = \{i : a^i x^* = b_i\}$. For more detail on LP duality and sensitivity, refer to any standard linear optimization text, such as [3, Chapters 4 and 5].

	Region Counts			Times Best is Optimal		
Problem	C^-	$C^+ \setminus C^-$	$\mathbb{R}^n \setminus C^+$	$C^+ \setminus C^-$	$\mathbb{R}^n \setminus C^+$	Total
`dcmulti`	9	43	948	43	278	330
`egout`	0	15	985	8	271	279
`gen`	0	0	1000	-	0	0
`khb05250`	0	5	995	3	55	58
`lseu`	0	0	1000	-	608	608
`mod008`	0	0	1000	-	4	4
`modglob`	0	401	599	393	438	831
`p0033`	0	0	1000	-	3	3
`p0201`	0	0	1000	-	699	699
`p0282`	0	1	999	0	664	664
`pp08a`	0	0	1000	-	41	41
`qiu`	0	0	1000	-	23	23
`stein27`	0	0	1000	-	0	0
`vpm2`	0	0	1000	-	23	23

Table 10
Region and optimality counts for 20% perturbations.

For our next set of experiments, we have chosen instances of the assignment problem, which can be formulated as a binary integer program with a totally unimodular constraint matrix. We generate four 20×20 specific problem instances using costs from problem `assign100`, originally from [1]. This problem can be found as part of the OR library [2].

For each instance, we examine three separate cases for the set of variables under scrutiny:

i) The variables in the top left 5×5 sub-matrix are under scrutiny.
ii) The variables in the top left 10×10 sub-matrix are under scrutiny.
iii) The variables in the entire 20×20 matrix are under scrutiny.

We chose instances of these sizes because, for the MIPLIB instances used in our earlier experiments, the average number of variables is about 400 and the average number of binaries is about 100. Thus the instances in (*ii*) are of comparable size. The instances in (*i*) and (*iii*) allow us to see how the results change as a function of the number of binaries under scrutiny.

As in the previous section, for each experiment we first solve the original

	$\frac{(\text{optimal}-\text{best})}{\lvert\text{optimal}\rvert}$		95% CI on Relative Diff.	
Problem	Average	St. Dev.	LB	UB
`dcmulti`	5.36E-06	3.71E-06	5.36E-06	5.37E-06
`egout`	2.37E-03	2.75E-03	2.36E-03	2.37E-03
`gen`	7.78E-04	2.56E-04	7.78E-04	7.79E-04
`khb05250`	2.60E-03	1.67E-03	2.59E-03	2.60E-03
`lseu`	2.53E-03	4.40E-03	2.52E-03	2.53E-03
`mod008`	4.91E-02	1.93E-02	4.91E-02	4.92E-02
`modglob`	3.18E-06	9.95E-06	3.16E-06	3.20E-06
`p0033`	2.19E-02	7.63E-03	2.19E-02	2.19E-02
`p0201`	9.72E-04	2.13E-03	9.68E-04	9.76E-04
`p0282`	4.97E-04	1.26E-03	4.94E-04	4.99E-04
`pp08a`	6.90E-03	4.95E-03	6.89E-03	6.91E-03
`qiu`	1.02E-01	5.19E-02	1.02E-01	1.02E-01
`stein27`	2.67E-02	7.51E-03	2.67E-02	2.67E-02
`vpm2`	1.28E-02	7.20E-03	1.27E-02	1.28E-02

Table 11
Relative difference results for 20% perturbations.

instance and complement variables to get $x^* = 0$ as an optimal solution, generate C as the cone of active constraints, and use Algorithm 1 to compute an inner (C^-) and outer (C^+) approximation of C. We then generate a random direction vector d, where each component $d_i \backsim U(0,1)$ is an i.i.d. random variable. We calculate the value

$$\lambda^- = \max\{\lambda : c^* + \lambda d \in C^-\},$$

and similarly define and calculate λ^* and λ^+ for C and C^+, respectively. Note that we always have the relation $\lambda^- \leq \lambda^* \leq \lambda^+$, because $C^- \subseteq C \subseteq C^+$. In addition, we compute the following statistics:

- The ratio $\frac{\lambda^*-\lambda^-}{\lambda^+-\lambda^-}$ as an indicator of how much of the uncertainty is caused by underapproximation of C^-.
- The number of times $\lambda^* - \lambda^- > \lambda^+ - \lambda^*$; that is, the number of times the majority of the uncertainty encountered in the direction d is caused by an under-approximation of C^-.
- The number of times $\lambda^- = \lambda^+$; that is, the number of times C^- and C^+ predict C exactly along the direction d.

For each assignment instance, we examine 100,000 randomly generated d vectors. Table 12 provides the averages of λ^-, λ^*, λ^+, $\frac{\lambda^*-\lambda^-}{\lambda^+-\lambda^-}$, while Table 13 has the number of times $\lambda^* - \lambda^- > \lambda^+ - \lambda^*$ and $\lambda^- = \lambda^+$, both obtained from the 100,000 generated vectors for each instance.

Problem	λ^- (avg.)	λ^* (avg.)	λ^+ (avg.)	$\frac{\lambda^*-\lambda^-}{\lambda^+-\lambda^-}$ (avg.)
5×5				
assign20_1	6.0086	12.1997	12.2018	0.9999
assign20_2	4.2159	7.3348	7.3355	0.9999
assign20_3	5.2905	6.3897	6.4002	0.9975
assign20_4	3.8208	4.0776	4.0780	0.9997
10×10				
assign20_1	1.2708	2.3431	2.3477	0.9982
assign20_2	1.5802	2.8811	2.8811	1.0000
assign20_3	1.5977	1.8501	1.8501	1.0000
assign20_4	1.4261	3.2234	3.2802	0.9789
20×20				
assign20_1	0.3669	0.9154	0.9246	0.9858
assign20_2	0.4758	0.7647	0.7647	1.0000
assign20_3	0.2593	0.2613	0.2613	1.0000
assign20_4	0.2397	0.3493	0.3494	0.9999

Table 12
Averages for the assignment instances.

In almost every single case where C^- and C^+ differ in the random direction d, we have $\lambda^* - \lambda^- > \lambda^+ - \lambda^*$, which indicates that more of the uncertainty region is caused by under-approximation of C^-. In fact, the average data for the ratio $\frac{\lambda^*-\lambda^-}{\lambda^+-\lambda^-}$ suggests that this under-approximation is almost solely responsible for the uncertainty. Together, both facts indicate that the region C^- defined by Theorem 3.3 may be too conservative when C^- and C^+ differ significantly.

Nevertheless, not all the results point to under-approximation. For example, with problem `assign20_3` we find that in any of the three scenarios the two approximations C^- and C^+ match exactly in over half of the random directions. In fact, this number jumps to over 92% in the experiment in which all 400 variables are under scrutiny. This implies that the approximations match the stability region C exactly in these directions.

We note that the results for the instances in which the costs change for 25 variables are significantly better than those for which the costs change for 100 and 400 variables. However, there is not much difference between the

Problem	$\lambda^* - \lambda^- > \lambda^+ - \lambda^*$ (ct.)	$\lambda^- = \lambda^+$ (ct.)
5×5		
`assign20_1`	62,835	37,164
`assign20_2`	98,325	1,674
`assign20_3`	23,673	76,326
`assign20_4`	16,899	83,100
10×10		
`assign20_1`	99,995	4
`assign20_2`	99,999	0
`assign20_3`	45,613	54,386
`assign20_4`	99,996	2
20×20		
`assign20_1`	100,000	0
`assign20_2`	100,000	0
`assign20_3`	7,614	92,386
`assign20_4`	98,298	1,702

Table 13
Counts for the assignment instances.

results with cost changes for 100 and 400 variables, so it is hard to draw conclusions about the quality of results with respect to the percentage of variables whose costs change. Finally, although we have not shown the details, as in the MIPLIB experiments, the set of solutions produced by Algorithm 1 almost always contains an optimal solution for the new costs.

6 Conclusions and Further Research

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 show that the inner and outer neighborhoods we obtain closely approximate the true stability region. In addition, the list of solutions generated as a byproduct of our algorithm can be used effectively to produce a high-quality solution for a true cost vector that is close to our estimate c^*.

Although the algorithm we have developed is flexible and accommodates a very general optimization problem, this same versatility implies that we sacrifice information when dealing with some types of problems. Specifically, consider the pure binary linear case, where $X \subseteq \{0,1\}^n$ and $P(c^*)$ becomes

$$\begin{aligned} \max\ & c^* x \\ \text{s.t. } & x \in X. \end{aligned}$$

The stability region of $x^* = 0$ is given by

$$C = \{c \in \mathbb{R}^n : cx \leq 0, \forall\ x \in X\}.$$

That is, C is a cone defined by at most $|X|$ inequalities with 0-1 coefficients. If we run Algorithm 1 on a pure binary problem, the inner neighborhood may cut off the origin (as in Example 2a,) even though the origin belongs to any pure binary stability region. A specialized approach may be able to exploit the additional conic structure to construct a larger inner neighborhood. As the pure binary program is an important special case, we plan to investigate it further.

An unexpected issue arose in solving the constrained instances $P(c^*)$. For certain instances, the solve time increased dramatically, peaking at times several orders of magnitude greater than for the original instance. Adding the cut $\sum_{i \in I} x_i \geq 1$ to all leaf nodes of the branch-and-bound tree and continuing from there might be more efficient than adding the cut and then resolving from scratch.

Acknowledgement

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

References

[1] J. E. Beasley, Linear programming on Cray supercomputers, Journal of the Operational Research Society 41 (1990) 133–139.

[2] J. E. Beasley, OR-Library: distributing test problems by electronic mail, Journal of the Operational Research Society 41 (11) (1990) 1069–1072, library available on-line at `http://people.brunel.ac.uk/~mastjjb/jeb/info.html`.

[3] D. Bertsimas, J. N. Tsitsiklis, Introduction to Linear Optimization, 1st ed., Athena Scientific, Belmont, Massachusetts, 1997.

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

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

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

[7] A. Crema, An algorithm for the multiparametric 0-1-integer linear programming problem relative to the objective function, European Journal of Operational Research 125 (1) (2000) 18–24.

[8] D. Ghosh, G. Sierksma, On the complexity of determining tolerances for ε-optimal solutions to min-max combinatorial optimization problems, Research Report 00A35, University of Groningen, Research Institute SOM (Systems, Organisations and Management), available on-line at `http://ideas.repec.org/p/dgr/rugsom/00a35.html` (2000).

[9] M. Hifi, H. Mhalla, 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) (2005) 239–260.

[10] S. Holm, D. Klein, Three methods for postoptimal analysis in integer linear programming, Mathematical Programming Study 21 (1984) 97–109.

[11] C. Kenyon, M. Sellmann, Plan B: Uncertainty/Time Trade-Offs for Linear and Integer Programming, working paper.

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

[13] R. Ramaswamy, N. Chakravarti, D. Ghosh, Complexity of determining exact tolerances for min-max combinatorial optimization problems, Research Report 00A22, University of Groningen, Research Institute SOM (Systems, Organisations and Management), available on-line at `http://ideas.repec.org/p/dgr/rugsom/00a22.html` (2000).

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

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

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

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

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