2000

Valid Integer Polytope (VIP) Penalties for Branch-and-Bound Enumeration

Brown, Gerald G.

þÿBrown, G.G., Dell, R.F., and Olson, M.P., 2000, Valid Integer Polytope
þÿBranch-and-Bound Enumeration, Operations Research Letters, 26, pp. 1

ELSEVIER

Operations Research Letters 26 (2000) 117–126

operations research letters

www.elsevier.com/locate/orms

Valid integer polytope (VIP) penalties for branch-and-bound enumeration

Gerald G. Brown[a], Robert F. Dell[a, *], Michael P. Olson[b]

[a]Operations Research Department, Naval Postgraduate School, Monterey, CA 93943-5219, USA
[b]INSIGHT, Inc., Sudley North Business Center, 7960 Donegan Drive, Suite 233, Manassas, VA 20109, USA

Received 1 December 1998; received in revised form 1 October 1999

Abstract

We introduce new penalties, called *valid integer polytope* (*VIP*) penalties, that tighten the bound of an integer-linear program during branch-and-bound enumeration. Early commercial codes for branch and bound commonly employed penalties developed from the dual simplicial lower bound on the cost of restricting fractional integer variables to proximate integral values. VIP penalties extend and tighten these for ubiquitous k-pack, k-partition, and k-cover constraints. In real-world problems, VIP penalties occasionally tighten the bound by more than an order of magnitude, but they usually offer small bound improvement. Their ease of implementation, speed of execution, and occasional, overwhelming success make them an attractive addition during branch-and-bound enumeration. © 2000 Elsevier Science B.V. All rights reserved.

Keywords: Integer-linear programming; Branch and bound; Penalties

1. Preliminaries

The integer linear program (ILP) is defined here as

$$\begin{array}{lll} \text{ILP} & \text{minimize} & \boldsymbol{cx} \\ & \text{subject to} & \boldsymbol{Ax} = \boldsymbol{b}, \quad \boldsymbol{x} \geqslant \boldsymbol{0}, \qquad (1) \\ & & x_j \in \{0,1\} \quad \forall j \in R, \qquad (\text{BINARY}) \end{array}$$

where $\boldsymbol{c}$ and $\boldsymbol{x}$ are n-vectors, $\boldsymbol{b}$ is an m-vector, $\boldsymbol{A}$ is an $m \times n$-matrix, and R is an index set of binary variables. Let $\tilde{\boldsymbol{x}}$ be an admissible incumbent solution to ILP, i.e., $\tilde{\boldsymbol{x}}$ satisfies (1) and (BINARY), and let $v(\tilde{\boldsymbol{x}}) \equiv \boldsymbol{c}\tilde{\boldsymbol{x}}$.

* Corresponding author.

Define the continuous relaxation of ILP as

$$\begin{aligned}
\text{LP} \quad & \text{minimize} \quad && \boldsymbol{cx} \\
& \text{subject to} && \boldsymbol{Ax} = \boldsymbol{b}, \quad \boldsymbol{x} \geqslant \boldsymbol{0}, \\
& && x_j \in [0,1] \quad \forall j \in R, \qquad \text{(CSR)}
\end{aligned}$$

where (CSR) is the continuous simplicial relaxation of (BINARY). Let $\hat{\boldsymbol{x}}$ be optimal to LP. Thus, $v(\hat{\boldsymbol{x}}) \equiv \boldsymbol{c}\hat{\boldsymbol{x}} \leqslant v(\tilde{\boldsymbol{x}})$. Solving LP using the simplex method, we obtain $\hat{\boldsymbol{x}}$ and a revised representation of LP in nonbasic variable space:

$$\begin{aligned}
\text{LPN} \quad & \text{minimize} \quad && v(\hat{\boldsymbol{x}}) + \sum_{j \in N} \hat{c}_j \hat{x}_j \\
& \text{subject to} && x_i = \hat{x}_i - \sum_{j \in N} \hat{a}_{ij} \hat{x}_j \quad \forall i \notin N, \\
& && x_j \geqslant 0 \quad \forall j, \\
& && x_j \in [0,1] \quad \forall j \in R,
\end{aligned}$$

where N is the index set of nonbasic variables, $\hat{a}_{ij}$ is the coefficient of nonbasic variable x_j in the representation of basic variable x_i, and $\hat{c}_j$ is the reduced cost of nonbasic variable x_j.

For solution $\hat{\boldsymbol{x}}$ to LP, define F as the index set of variables fractional in LP and restricted by (BINARY) in ILP: $F = \{j \in R\colon \hat{x}_j \in (0,1)\}$. If LP yields a solution with $0 < \hat{x}_j < 1$ for some j (i.e., $F \neq \emptyset$), we can use a simplicial restriction to try to improve our lower bound on $v(\tilde{\boldsymbol{x}})$ by restricting x_j to a binary value. Define such a simplicial restriction for $j \in R$:

$$\begin{aligned}
\text{LPX}(\delta_j) \quad & \text{minimize} \quad && \boldsymbol{cx} \\
& \text{subject to} && \boldsymbol{Ax} = \boldsymbol{b}, \quad \boldsymbol{x} \geqslant \boldsymbol{0}, \\
& && x_{j'} \in [0,1] \quad \forall j' \in R \setminus \{j\}, \\
& && x_j = \delta_j,
\end{aligned}$$

where $\delta_j \in \{0,1\}$. Thus, $v(\hat{\boldsymbol{x}}) \leqslant \min\{v(\hat{\boldsymbol{x}}|\delta_j = 0),\ v(\hat{\boldsymbol{x}}|\delta_j = 1)\} \leqslant v(\tilde{\boldsymbol{x}})$ where $\hat{\boldsymbol{x}}|\delta_j$ indicates an optimal solution to LPX(δ_j).

Branch-and-bound enumeration employs restrictions of this sort to find admissible ILP solutions, or to establish an improved lower bound on the solutions to ILP that might be achieved via further restrictions. If such a lower bound is known to apply to all such restrictions, it improves the lower bound on the best-known incumbent solution to ILP.

Before solving LPX(δ_j), we can use remnants of the simplicial solution of LP to conservatively forecast the penalty of the LPX(δ_j) restriction (e.g., [4,10]). This forecast improves the lower bound on the best new incumbent we might find by solving LPX(δ_j), or any additional restrictions beyond those of LPX(δ_j).

Because solving problems like LPX(δ_j) can be difficult, and because there can be many to solve, prior bounds on such solutions can be of considerable value to the overall success of branch and bound. This helps motivate the early use of penalties constructed from remnants of the simplicial solution. But, as Nemhauser and Wolsey [6, p. 360] state in the context of branch selection

> They [Penalties] were used in early commercial codes but are not in favor now because they are too costly to compute relative to the value of the information they give.

While we do not disagree that exact penalties serve as a poor criterion for branch choice, we contend that exact penalties can be valuable for improving lower bounds for difficult ILPs, and conjecture that some commercial codes do employ penalties to improve lower bounds on incumbent solutions. We lament that such details are proprietary secrets for most current commercial codes. We show that these penalties can be specialized and strengthened in the presence of k-pack, k-cover, and k-partition constraints, and are not costly to compute relative to their occasional stunning bound improvements.

2. VIP penalties for k-pack, k-cover, and k-partition

Define $\underline{\theta}_j$ ($\underline{\theta}_j \leqslant v(\hat{\boldsymbol{x}}|\delta_j = 0) - v(\hat{\boldsymbol{x}})$) as the "simplicial down-penalty" associated with $j \in R$ and $\bar{\theta}_j$ ($\bar{\theta}_j \leqslant v(\hat{\boldsymbol{x}}|\delta_j = 1) - v(\hat{\boldsymbol{x}})$) as the "simplicial up-penalty". These "up and down" penalties derive from the simplicial representation of a solution to LP, and are readily computable from computational remnants of that solution. Tomlin [10] calculates these penalties for $i \in R$ as

$$\underline{\theta}_i = 0 \text{ for } \hat{x}_i = 0, \quad \underline{\theta}_i = \min_{j \in N, \hat{a}_{ij} > 0} \left\{ \begin{array}{ll} \hat{c}_j \left(\max\left\{ 1, \dfrac{\hat{x}_i}{\hat{a}_{ij}} \right\} \right) & \text{if } j \in R, \\ \hat{c}_j \left(\left\{ \dfrac{\hat{x}_i}{\hat{a}_{ij}} \right\} \right) & \text{if } j \notin R \end{array} \right\} \text{ for } 0 < \hat{x}_i \leqslant 1,$$

$$\bar{\theta}_i = \hat{c}_i \text{ for } i \in N, \quad \bar{\theta}_i = \min_{j \in N, \hat{a}_{ij} < 0} \left\{ \begin{array}{ll} \hat{c}_j \left(\max\left\{ 1, \dfrac{1-\hat{x}_i}{-\hat{a}_{ij}} \right\} \right) & \text{if } j \in R, \\ \hat{c}_j \left(\left\{ \dfrac{1-\hat{x}_i}{-\hat{a}_{ij}} \right\} \right) & \text{if } j \notin R, \end{array} \right\} \text{ for } i \notin N,\ 0 \leqslant \hat{x}_i < 1,$$

and $\bar{\theta}_i = 0$ for $\hat{x}_i = 1$.

A valid-integer-polytope (VIP) constraint is a valid inequality on the ILP integer polytope. The VIP constraints $0 \leqslant x_j \leqslant 1$ for $j \in R$ improve $v(\hat{\boldsymbol{x}})$ by $\min\{\underline{\theta}_j, \bar{\theta}_j\}$ (e.g., see [4, p. 120]). Because each variable $j \in F$ must be restricted to zero or one to satisfy (BINARY) in ILP, the collective improvement is

$$\theta(\text{CSR}) = \max_{j \in R}\{\min\{\underline{\theta}_j, \bar{\theta}_j\}\} = \max_{j \in F}\{\min\{\underline{\theta}_j, \bar{\theta}_j\}\}. \tag{2}$$

We strengthen the penalty offered by (2) for k-pack, k-cover, and k-partition constraints (original ILP constraints or valid inequalities to ILP). Tomlin [10] also strengthens the penalty offered by (2); he presents penalties deduced from the observation that the fractional source variable in a Gomory cut induces movement in the other variables from which a penalty can be derived (e.g., [7, pp. 192–193]). Because our development requires that we associate specific up and down penalties with particular variables, we cannot apply this result.

To motivate our development, we start with a simple pairwise 1-pack VIP constraint:

$$x_j + x_{j'} \leqslant 1, \quad \{j, j'\} = R' \subseteq R.$$

Here, we see that either x_j or $x_{j'}$ must be restricted to zero in ILP. Thus, we know besides the contribution of these two variables to $\theta(\text{CSR})$, the smaller of their two down penalties ($\min\{\underline{\theta}_j, \underline{\theta}_{j'}\}$) also applies. Combined with (2), we obtain

$$\max\{\min\{\underline{\theta}_j, \underline{\theta}_{j'}\}, \max\{\min\{\underline{\theta}_j, \bar{\theta}_j\}, \min\{\underline{\theta}_{j'}, \bar{\theta}_{j'}\}\}\}. \tag{3}$$

To generalize this observation, define $[l]_\mathrm{d}$ ($[l]_\mathrm{u}$) as the index $j \in R'$ ($j' \in R'$) corresponding to the lth ascending down-penalty (ascending up-penalty) in the set $R' \subseteq R$ and let $r' = |R'|$. That is, $R' = \{[1]_\mathrm{d}, [2]_\mathrm{d}, \dots, [r']_\mathrm{d}\} = \{[1]_\mathrm{u}, [2]_\mathrm{u}, \dots, [r']_\mathrm{u}\}$ where $\underline{\theta}_{[1]_\mathrm{d}} \leqslant \underline{\theta}_{[2]_\mathrm{d}} \leqslant \cdots \leqslant \underline{\theta}_{[r']_\mathrm{d}}$ and $\bar{\theta}_{[1]_\mathrm{u}} \leqslant \bar{\theta}_{[2]_\mathrm{u}} \leqslant \cdots \leqslant \bar{\theta}_{[r']_\mathrm{u}}$. Also define the ordered subsets of R', $R'_{k_\mathrm{d}} = \{[1]_\mathrm{d}, [2]_\mathrm{d}, \dots, [k]_\mathrm{d}\}$ and $R'_{k_\mathrm{u}} = \{[1]_\mathrm{u}, [2]_\mathrm{u}, \dots, [k]_\mathrm{u}\}$.

Using this notation, we know $\underline{\theta}_{[1]_\mathrm{d}} \leqslant \underline{\theta}_{[2]_\mathrm{d}}$ and $\underline{\theta}_{[1]_\mathrm{d}} \geqslant \min\{\underline{\theta}_{[1]_\mathrm{d}}, \bar{\theta}_{[1]_\mathrm{d}}\}$ so (3) becomes

$$\max\{\underline{\theta}_{[1]_\mathrm{d}}, \min\{\underline{\theta}_{[2]_\mathrm{d}}, \bar{\theta}_{[2]_\mathrm{d}}\}\}.$$

Property 1. *A general k*-pack *VIP constraint with $r' > k$:*

$$\sum_{j \in R'} x_j \leqslant k, \quad (k\text{-pack})$$

offers improvement of $v(\hat{\boldsymbol{x}})$ by

$$\theta(k\text{-pack}) = \max\left\{\underline{\theta}_{[r'-k]_\mathrm{d}}, \max_{j \in R' \setminus R'_{(r'-k)_\mathrm{d}}} \{\min\{\underline{\theta}_j, \bar{\theta}_j\}\}\right\}. \tag{4}$$

Proof. The term $\underline{\theta}_{[r'-k]_\mathrm{d}}$ arises from the $r'-k$ variables in set R' that must have value zero in ILP. The latter term derives from (2) and the result that

$$\underline{\theta}_{[r'-k]_\mathrm{d}} \geqslant \max_{j \in R'_{(r'-k)_\mathrm{d}}} \{\min\{\underline{\theta}_j, \bar{\theta}_j\}\}. \qquad \square$$

Property 2. *A k*-cover *VIP constraint with $r' > k$:*

$$\sum_{j \in R'} x_j \geqslant k, \quad (k\text{-cover})$$

offers improvement of $v(\hat{\boldsymbol{x}})$ by

$$\theta(k\text{-cover}) = \max\left\{\bar{\theta}_{[k]_\mathrm{u}}, \max_{j \in R' \setminus R'_{k_\mathrm{u}}} \{\min\{\underline{\theta}_j, \bar{\theta}_j\}\}\right\}. \tag{5}$$

Proof. The term $\bar{\theta}_{[k]_\mathrm{u}}$ is justified because at least k variables in set R' must have value one in ILP. The latter term derives from (2) and the result that

$$\bar{\theta}_{[k]_\mathrm{u}} \geqslant \max_{j \in R'_{k_\mathrm{u}}} \{\min\{\underline{\theta}_j, \bar{\theta}_j\}\}. \qquad \square$$

Property 3. *A k*-partition *VIP constraint with $r' > k$:*

$$\sum_{j \in R'} x_j = k, \quad (k\text{-partition})$$

offers improvement to $v(\hat{\boldsymbol{x}})$ by the maximum of the k-pack *and k*-cover *VIP penalties:*

$$\theta(k\text{-partition}) = \max\{\theta(k\text{-pack}), \theta(k\text{-cover})\}. \tag{6}$$

3. Pack and cover symmetry

There is a symmetry between the pack and cover VIP constraints and the associated VIP penalties presented here. Consider the k-pack constraint $\sum_{j \in R'} x_j \leqslant k$ and let $x_j = 1 - x'_j$ (a complement reflection) $\forall j \in R'$.

Then

$$((k\text{-pack) in } \boldsymbol{x}) \Leftrightarrow \sum_{j\in R'} x_j \leqslant k \Leftrightarrow \sum_{j\in R'}(1-x'_j) \leqslant k \Leftrightarrow \sum_{j\in R'} x'_j \geqslant r'-k \Leftrightarrow ((r'-k\text{-cover) in } \boldsymbol{x}').$$

Because $\underline{\theta}_j = \bar{\theta}'_j$, and $\bar{\theta}_j = \underline{\theta}'_j$, the VIP penalty for the k-pack is identical to that of the complement $r'-k$-cover:

$$\theta(k\text{-pack}) = \max\left\{ \underline{\theta}_{[r'-k]_{\mathrm{d}}}, \max_{j\in R'\backslash R'_{(r'-k)_{\mathrm{d}}}} \{\min\{\underline{\theta}_j, \bar{\theta}_j\}\} \right\},$$

$$\theta(r'-k\text{-cover}) = \max\left\{ \bar{\theta}'_{[r'-k]_{\mathrm{u}}}, \max_{j\in R'\backslash R'_{(r'-k)_{\mathrm{u}}}} \{\min\{\underline{\theta}'_j, \bar{\theta}'_j\}\} \right\}.$$

4. When VIP θ(k-pack) and θ(k-cover) can be stronger than θ(CSR)

The following establishes necessary conditions for which the VIP penalties $\theta(k$-pack) or $\theta(k$-cover) can be stronger than the conventional θ(CSR) penalty.

Define $\underline{R}' = \{j \in R'\colon \hat{x}_j = 0\}(\bar{R}' = \{j \in R'\colon \hat{x}_j = 1\})$ as the set of variables restricted (Binary) in ILP and equal to zero (one) in LP. Also define $F' = \{j \in R'\colon \hat{x}_j \in (0,1)\}$, $\underline{r}' = |\underline{R}'|$, $\bar{r}' = |\bar{R}'|$, and $f' = |F'|$. (Note: $R' = \underline{R}' \cup \bar{R}' \cup F'$ and $r' = \underline{r}' + \bar{r}' + f'$.)

Clearly, if $F' = \emptyset$ then $\theta(\text{CSR}) = \theta(k\text{-pack}) = \theta(k\text{-cover}) = 0$ in (2), (4), and (5) (i.e., the bounds provided by (2), (4), and (5) can be nonzero only when $f' \geqslant 1$).

From the definition of $\underline{R}'$, $\bar{R}'$, $\underline{\theta}_{[r'-k]_{\mathrm{d}}}$, and $\bar{\theta}_{[k]_{\mathrm{u}}}$ it follows that the value of $\theta(k$-pack) ($\theta(k$-cover)) can exceed the value of θ(CSR) only if $\underline{r}' \leqslant r' - (k+1)$ in k-pack ($\bar{r}' \leqslant k-1$ in k-cover).

Property 4. *The relationship between* $\underline{r}'$, $\bar{r}'$, *and* f'.

- $\underline{r}' \leqslant r' - (k+1)$ *in* k-pack ($\bar{r}' \leqslant k-1$ *in* k-cover) *implies* $f' \geqslant 1$.

Proof. $r' = \underline{r}' + \bar{r}' + f'$ or $f' = r' - \underline{r}' - \bar{r}'$ and $\bar{r}' \leqslant k$ in k-pack ($\underline{r}' \leqslant r' - k$ in k-cover) providing $f' \geqslant r' - r' - (k+1) - k(f' \geqslant r' - (r'-k) - (k-1))$ or $f' \geqslant 1$. □

- $f' \geqslant 1$ *does not necessarily imply* $\underline{r}' \leqslant r' - (k+1)$ *in* k-pack ($\bar{r}' \leqslant k-1$ *in* k-cover) *when* $k \geqslant 2$ ($k \leqslant r'-2$). For example, $x_1 + x_2 + x_3 \leqslant 2$ with $\hat{x}_1 = 0$, $\hat{x}_2 = \hat{x}_3 = 0.5$ provides $\underline{r}' = 1$ and $f' = 2$ ($x_1 + x_2 + x_3 \geqslant 1$ with $\hat{x}_1 = 1$, $\hat{x}_2 = \hat{x}_3 = 0.5$ provides $\bar{r}' = 1$ and $f' = 2$).
- *When* $k \leqslant 1$ ($k \geqslant r'-1$), $f' > 1$ *implies* $\underline{r}' \leqslant r' - (k+1)$ ($\bar{r}' \leqslant k-1$) *in* k-pack (k-cover).

Proof. For k-pack, assume $\underline{r}' > r' - (k+1) \geqslant r' - 2$ or $2 > r' - \underline{r}'$ implying (since $f' = r' - \underline{r}' - \bar{r}'$) the contradiction $f' \leqslant 1$. For k-cover, assume $\bar{r}' > k - 1 \geqslant r' - 2$ or $2 > r' - \bar{r}'$ implying, again, the contradiction $f' \leqslant 1$. □

It is therefore necessary for $\underline{r}' \leqslant r' - (k+1)$ in k-pack ($\bar{r}' \leqslant k-1$ in k-cover) for the value of $\theta(k$-pack) ($\theta(k$-cover)) to exceed the value of θ(CSR). When $f' > 1$ this necessary condition is satisfied for $k \leqslant 1$ in k-pack ($k \geqslant r'-1$ in k-cover) but not for arbitrary k.

5. VIP opportunities abound in real-world models

Pack, cover, and partition constraints abound in real-world models. Generalized upper bound (GUB) constraints [3] are often present in large numbers (see e.g. [1] and references therein). We submit that GUB constraints often apply to sets of binary variables, and that the coefficients of these constraints are often one, or can be rendered so by scaling and reflection. Other row-factorizations (e.g., pure network rows) also exhibit such structure. Furthermore, the right-hand side of such integer forms can be tightened to some integer k after scaling and reflection of row coefficients. The most frequent value of k is one.

A special ordered set (SOS) constraint of Type I [9] is a 1-pack constraint, $\sum_{j\in R'} x_j \leqslant 1$. Tomlin suggests branching by partitioning the variables in R' into two subsets, each of which is alternately forced to zero, inducing a 0-pack constraint ($\sum_{j\in R_1} x_j \leqslant 0$ for $R_1 \subset R'$).

Both GUB and SOS constraints are potential sources for VIP penalties when their incident binary variables are fractional.

A "simple variable upper bound constraint" or "SVUB constraint" has the form

$$x_j \leqslant x_{j'}, \quad \{j, j'\} \subseteq R, \quad \text{(SVUB)}$$

and is a simple example of a variable upper bound [8]. SVUB can be used to control binary x_j with binary $x_{j'}$: "j only if j'." Such constraints occur frequently, and when a model exhibits SVUBs, it often contains many of them.

Proposition 5. *Using the* SVUB *constraint*, $v(\hat{\mathbf{x}})$ *may be improved by*

$$\theta(\text{SVUB}) = \min\{\max\{\underline{\theta}_j, \underline{\theta}_{j'}\}, \max\{\bar{\theta}_{j'}, \min\{\underline{\theta}_j, \bar{\theta}_j\}\}\}. \tag{7}$$

Proof. If $x_{j'} = 0$ then $x_j = 0$ and $\max\{\underline{\theta}_j, \underline{\theta}_{j'}\}$ is a valid bound. If $x_{j'} = 1$ then $x_j = 0$ or 1 and $\max\{\bar{\theta}_{j'}, \min\{\underline{\theta}_j, \bar{\theta}_j\}\}$ is a valid bound. Because $x_{j'}$ may take on the value zero or one, we must take the minimum of these two bounds. □

Although not immediately obvious, (7) is equivalent to the result obtained when a SVUB constraint is transformed to an equivalent 1-pack constraint by complementing $x_{j'}$:

$$x_j \leqslant x_{j'} \Leftrightarrow x_j \leqslant 1 - x'_{j'} \Leftrightarrow x_j + x'_{j'} \leqslant 1,$$

providing under (4)

$$\max\{\min\{\underline{\theta}_j, \underline{\theta}'_{j'} = \bar{\theta}_{j'}\}, \max\{\min\{\underline{\theta}_j, \bar{\theta}_j\}, \min\{\underline{\theta}_{j'}, \bar{\theta}_{j'}\}\}\}$$

or

$$x_j \leqslant x_{j'} \Leftrightarrow 1 - x'_j \leqslant x_{j'} \Leftrightarrow x'_j + x_{j'} \geqslant 1,$$

providing the same improvement under (5)

$$\max\{\min\{\bar{\theta}'_j = \underline{\theta}_j, \bar{\theta}_{j'}\}, \max\{\min\{\underline{\theta}_j, \bar{\theta}_j\}, \min\{\underline{\theta}_{j'}, \bar{\theta}_{j'}\}\}\}. \tag{8}$$

Proposition 6. *Eqs.* (7) *and* (8) *are equivalent.*

Proof. We establish the equivalence by looking at all partial orderings of $\underline{\theta}_j$, $\bar{\theta}_j$, $\underline{\theta}_{j'}$, and $\bar{\theta}_{j'}$. All partial orderings must have $\underline{\theta}_j \leqslant \underline{\theta}_{j'}$ (if $x_{j'}$ becomes zero, then x_j must become zero) and $\bar{\theta}_j \geqslant \bar{\theta}_{j'}$ (if x_j becomes one, then $x_{j'}$ must become one). The following is an exhaustive list of remaining partial orders and the result obtained in both (7) and (8):

- $\underline{\theta}_j \leqslant \bar{\theta}_j$, $\underline{\theta}_j \leqslant \bar{\theta}_{j'}$, $\underline{\theta}_{j'} \leqslant \bar{\theta}_j$, $\underline{\theta}_{j'} \leqslant \bar{\theta}_{j'}$ yielding $\underline{\theta}_{j'}$,
- $\underline{\theta}_j \leqslant \bar{\theta}_j$, $\underline{\theta}_j \leqslant \bar{\theta}_{j'}$, $\underline{\theta}_{j'} \leqslant \bar{\theta}_j$, $\underline{\theta}_{j'} \geqslant \bar{\theta}_{j'}$ yielding $\bar{\theta}_{j'}$,
- $\underline{\theta}_j \leqslant \bar{\theta}_j$, $\underline{\theta}_j \leqslant \bar{\theta}_{j'}$, $\underline{\theta}_{j'} \geqslant \bar{\theta}_j$, $\underline{\theta}_{j'} \leqslant \bar{\theta}_{j'}$ yielding $\underline{\theta}_{j'}$,
- $\underline{\theta}_j \leqslant \bar{\theta}_j$, $\underline{\theta}_j \leqslant \bar{\theta}_{j'}$, $\underline{\theta}_{j'} \geqslant \bar{\theta}_j$, $\underline{\theta}_{j'} \geqslant \bar{\theta}_{j'}$ yielding $\bar{\theta}_{j'}$,
- $\underline{\theta}_j \leqslant \bar{\theta}_j$, $\underline{\theta}_j \geqslant \bar{\theta}_{j'}$, $\underline{\theta}_{j'} \leqslant \bar{\theta}_j$, $\underline{\theta}_{j'} \leqslant \bar{\theta}_{j'}$ yielding $\underline{\theta}_j$, $\underline{\theta}_{j'}$, or $\bar{\theta}_{j'}$
  because $\underline{\theta}_{j'} \leqslant \bar{\theta}_{j'} \leqslant \underline{\theta}_j$ (but $\underline{\theta}_{j'} \geqslant \underline{\theta}_j$) or $\underline{\theta}_{j'} = \bar{\theta}_{j'} = \underline{\theta}_j$,
- $\underline{\theta}_j \leqslant \bar{\theta}_j$, $\underline{\theta}_j \geqslant \bar{\theta}_{j'}$, $\underline{\theta}_{j'} \leqslant \bar{\theta}_j$, $\underline{\theta}_{j'} \geqslant \bar{\theta}_{j'}$ yielding $\underline{\theta}_j$,
- $\underline{\theta}_j \leqslant \bar{\theta}_j$, $\underline{\theta}_j \geqslant \bar{\theta}_{j'}$, $\underline{\theta}_{j'} \geqslant \bar{\theta}_j$, $\underline{\theta}_{j'} \leqslant \bar{\theta}_{j'}$ yielding $\underline{\theta}_j$, $\bar{\theta}_j$, $\underline{\theta}_{j'}$, or $\bar{\theta}_{j'}$
  because $\bar{\theta}_{j'} \leqslant \underline{\theta}_j \leqslant \bar{\theta}_j \leqslant \underline{\theta}_{j'} \leqslant \bar{\theta}_{j'}$ or $\underline{\theta}_j = \bar{\theta}_j = \underline{\theta}_{j'} = \bar{\theta}_{j'}$,
- $\underline{\theta}_j \leqslant \bar{\theta}_j$, $\underline{\theta}_j \geqslant \bar{\theta}_{j'}$, $\underline{\theta}_{j'} \geqslant \bar{\theta}_j$, $\underline{\theta}_{j'} \geqslant \bar{\theta}_{j'}$ yielding $\underline{\theta}_j$,
- $\underline{\theta}_j \geqslant \bar{\theta}_j$, $\underline{\theta}_j \leqslant \bar{\theta}_{j'}$, $\underline{\theta}_{j'} \leqslant \bar{\theta}_j$, $\underline{\theta}_{j'} \leqslant \bar{\theta}_{j'}$ yielding $\underline{\theta}_j$, $\underline{\theta}_{j'}$, $\bar{\theta}_j$
  because $\underline{\theta}_{j'} \leqslant \bar{\theta}_j \leqslant \underline{\theta}_j$ (but $\underline{\theta}_{j'} \geqslant \underline{\theta}_j$) so $\underline{\theta}_{j'} = \bar{\theta}_j = \underline{\theta}_j$,
- $\underline{\theta}_j \geqslant \bar{\theta}_j$, $\underline{\theta}_j \leqslant \bar{\theta}_{j'}$, $\underline{\theta}_{j'} \leqslant \bar{\theta}_j$, $\underline{\theta}_{j'} \geqslant \bar{\theta}_{j'}$ yielding $\underline{\theta}_j$, $\bar{\theta}_j$, $\underline{\theta}_{j'}$, or $\bar{\theta}_{j'}$
  because $\bar{\theta}_j \leqslant \underline{\theta}_j \leqslant \bar{\theta}_{j'} \leqslant \underline{\theta}_{j'} \leqslant \bar{\theta}_j$ or $\underline{\theta}_j = \bar{\theta}_j = \underline{\theta}_{j'} = \bar{\theta}_{j'}$,
- $\underline{\theta}_j \geqslant \bar{\theta}_j$, $\underline{\theta}_j \leqslant \bar{\theta}_{j'}$, $\underline{\theta}_{j'} \geqslant \bar{\theta}_j$, $\underline{\theta}_{j'} \leqslant \bar{\theta}_{j'}$ yielding $\underline{\theta}_j$, $\bar{\theta}_j$, $\underline{\theta}_{j'}$, or $\bar{\theta}_{j'}$
  because $\bar{\theta}_j \leqslant \underline{\theta}_{j'} \leqslant \bar{\theta}_{j'}$ and $\bar{\theta}_j \leqslant \underline{\theta}_j \leqslant \bar{\theta}_{j'}$ (but $\bar{\theta}_j \geqslant \bar{\theta}_{j'}$) so $\underline{\theta}_j = \bar{\theta}_j = \underline{\theta}_{j'} = \bar{\theta}_{j'}$,
- $\underline{\theta}_j \geqslant \bar{\theta}_j$, $\underline{\theta}_j \leqslant \bar{\theta}_{j'}$, $\underline{\theta}_{j'} \geqslant \bar{\theta}_j$, $\underline{\theta}_{j'} \geqslant \bar{\theta}_{j'}$ yielding $\bar{\theta}_j$, $\underline{\theta}_{j'}$, or $\bar{\theta}_{j'}$
  because $\bar{\theta}_j \leqslant \underline{\theta}_j \leqslant \bar{\theta}_{j'}$ (but $\bar{\theta}_j \geqslant \bar{\theta}_{j'}$) so $\bar{\theta}_j = \underline{\theta}_j = \bar{\theta}_{j'}$,
- $\underline{\theta}_j \geqslant \bar{\theta}_j$, $\underline{\theta}_j \geqslant \bar{\theta}_{j'}$, $\underline{\theta}_{j'} \leqslant \bar{\theta}_j$, $\underline{\theta}_{j'} \leqslant \bar{\theta}_{j'}$ yielding $\underline{\theta}_j$, $\bar{\theta}_j$, $\underline{\theta}_{j'}$, or $\bar{\theta}_{j'}$
  because $\underline{\theta}_{j'} \leqslant \bar{\theta}_{j'} \leqslant \underline{\theta}_j$ and $\underline{\theta}_{j'} \leqslant \bar{\theta}_j \leqslant \underline{\theta}_j$ (but $\underline{\theta}_{j'} \geqslant \underline{\theta}_j$) so $\underline{\theta}_j = \bar{\theta}_j = \underline{\theta}_{j'} = \bar{\theta}_{j'}$,
- $\underline{\theta}_j \geqslant \bar{\theta}_j$, $\underline{\theta}_j \geqslant \bar{\theta}_{j'}$, $\underline{\theta}_{j'} \leqslant \bar{\theta}_j$, $\underline{\theta}_{j'} \geqslant \bar{\theta}_{j'}$ yielding $\underline{\theta}_j$, $\bar{\theta}_j$, or $\underline{\theta}_{j'}$
  because $\underline{\theta}_{j'} \leqslant \bar{\theta}_j \leqslant \underline{\theta}_j$ (but $\underline{\theta}_{j'} \geqslant \underline{\theta}_j$) so $\underline{\theta}_{j'} = \bar{\theta}_j = \underline{\theta}_j$,
- $\underline{\theta}_j \geqslant \bar{\theta}_j$, $\underline{\theta}_j \geqslant \bar{\theta}_{j'}$, $\underline{\theta}_{j'} \geqslant \bar{\theta}_j$, $\underline{\theta}_{j'} \leqslant \bar{\theta}_{j'}$ yielding $\underline{\theta}_j$, $\bar{\theta}_j$, $\underline{\theta}_{j'}$ or $\bar{\theta}_{j'}$
  because $\bar{\theta}_j \leqslant \underline{\theta}_{j'} \leqslant \bar{\theta}_{j'} \leqslant \underline{\theta}_j$ (but $\underline{\theta}_{j'} \geqslant \underline{\theta}_j$ and $\bar{\theta}_j \geqslant \bar{\theta}_{j'}$) so $\underline{\theta}_j = \underline{\theta}_{j'} = \bar{\theta}_j = \bar{\theta}_{j'}$,
- $\underline{\theta}_j \geqslant \bar{\theta}_j$, $\underline{\theta}_j \geqslant \bar{\theta}_{j'}$, $\underline{\theta}_{j'} \geqslant \bar{\theta}_j$, $\underline{\theta}_{j'} \geqslant \bar{\theta}_{j'}$ yielding $\bar{\theta}_j$. □

Consider a more general form of the SVUB constraint:

$$\sum_{j \in R'} x_j \leqslant k\, x_{j'}, \quad (k\text{-VUB})$$

where $k \geqslant |R'|$, and consider its equivalent form

$$x_j \leqslant x_{j'} \quad \forall j \in R'. \quad \text{(VUBS)}$$

"k-cardinality" constraints such as these appear frequently in real-world models.

We know (7) is valid for each $j \in R'$ in VUBS, providing

$$\max_{j \in R'} \{\min\{\max\{\underline{\theta}_j, \underline{\theta}_{j'}\}, \max\{\bar{\theta}_{j'}, \min\{\underline{\theta}_j, \bar{\theta}_j\}\}\}\}.$$

Since $\underline{\theta}_j \leqslant \underline{\theta}_{j'}$, for all $j \in R'$, this becomes

$$\max_{j \in R'} \{\min\{\underline{\theta}_{j'}, \max\{\bar{\theta}_{j'}, \min\{\underline{\theta}_j, \bar{\theta}_j\}\}\}\}$$

or

$$\min\left\{\underline{\theta}_{j'}, \max\left\{\bar{\theta}_{j'}, \max_{j \in R'}\{\min\{\underline{\theta}_j, \bar{\theta}_j\}\}\right\}\right\}. \tag{9}$$

Table 1

	x_1	x_2	x_3	x_4	x_5	x_6	s_1	s_2	s_3	s_4	RHS
z	0	0	0	−1	−1	−1	−1	−1	−1	0	3.0
x_1	1	0	0	1	0	0	−0.5	0.5	−0.5	0	0.5
x_2	0	1	0	0	1	0	−0.5	−0.5	0.5	0	0.5
x_3	0	0	1	0	0	1	0.5	−0.5	−0.5	0	0.5
s_4	0	0	0	1	1	1	−0.5	−0.5	−0.5	1	0.5

For the basic k-VUB constraint,

$$\sum_{j\in R'} x_j \leqslant k\, x_{j'} \Leftrightarrow \sum_{j\in R'} x_j \leqslant k\ (1-x'_{j'}) \Leftrightarrow \sum_{j\in R'} x_j + k\, x'_{j'} \leqslant k.$$

If $x'_{j'}=1$ ($x_{j'}=0$), then $\sum_{j\in R'} x_j=0$ and the resulting bound is $\max\{\underline{\theta}_{j'}, \underline{\theta}_{[k]_d}\}=\underline{\theta}_{j'}$. If $x'_{j'}=0$ ($x_{j'}=1$) then $x_j=0$ or 1 $\forall j\in R'$ and the resulting bound is $\max\{\bar{\theta}_{j'}, \max_{j\in R'}\{\min\{\underline{\theta}_j,\bar{\theta}_j\}\}\}$. Since $x_{j'}$ can have either value 0 or 1, the resulting bound (equivalent to (9)) is

$$\min\left\{\underline{\theta}_{j'}, \max\left\{\bar{\theta}_{j'}, \max_{j\in R'}\{\min\{\underline{\theta}_j,\bar{\theta}_j\}\}\right\}\right\}.$$

6. An illustrative example

Consider a simple ILP

$$\begin{aligned}
\text{minimize}\quad & z=2x_1+2x_2+2x_3+3x_4+3x_5+3x_6\\
\text{subject to}\quad & x_1+x_2+x_4+x_5\geqslant 1,\\
& x_2+x_3+x_5+x_6\geqslant 1,\\
& x_1+x_3+x_4+x_6\geqslant 1,\\
& x_1+x_2+x_3\geqslant 1,\\
& x_1,x_2,x_3\in\{0,1\}\quad x_4,x_5,x_6\geqslant 0.
\end{aligned}$$

An optimal solution is $\tilde{x}_1=\tilde{x}_2=1$ with $z=4$.

The explicit simplicial remnants of LP are given in Table 1. LP provides a lower bound on admissible z of 3.

Using the information in Table 1, we compute the down-penalties (e.g., [7, p. 191]) $\underline{\theta}_1=\underline{\theta}_2=\underline{\theta}_3=0.5$ and the up-penalties $\bar{\theta}_1=\bar{\theta}_2=\bar{\theta}_3=1$. For example, $\hat{x}_1=0.5-\hat{x}_4+0.5\hat{s}_1-0.5\hat{s}_2+0.5\hat{s}_3$ so

$$\underline{\theta}_1=\min\left\{\frac{0.5(1)}{1},\frac{0.5(1)}{0.5}\right\}=0.5$$

because the minimum penalty for lowering $\hat{x}_1$ to zero can be achieved by either increasing $\hat{x}_4$ by 0.5 (a penalty of 0.5(1)/1) or increasing $\hat{s}_2$ by 1.0 (a penalty of 0.5(1)/0.5).

$$\bar{\theta}_1=\min\left\{\frac{0.5(1)}{0.5},\frac{0.5(1)}{0.5}\right\}=1$$

because the minimum penalty for raising $\hat{x}_1$ to one can be achieved by either increasing $\hat{s}_1$ or $\hat{s}_3$ by 1.0 (a penalty of 0.5(1)/0.5).

$\theta(CSR)$ (Eq. (2)) is 0.5. Thus, without VIP penalties, the lower bound on admissible z is 3.5.

Apply (4) to the original 1-cover constraint $x_1 + x_2 + x_3 \geqslant 1$ and $\theta(\text{1-cover}) = 1$; this 1-cover VIP penalty raises the lower bound on z to 4, closing the integrality gap completely.

7. Implementation and anecdotal computational experience

We developed VIP penalties to help solve real-world, large-scale mixed-integer problems. We qualify their success with two caveats: the existing design of our solver (the X-system, e.g., [1]) influences our implementation, and our experience is limited to problems we frequently solve. Pursuant to our caveats, we do not clutter this paper with tables of our benchmarks because they would hardly constitute reproducible scientific experiments with other solvers.

Predominantly, we deal with binary set partitions, and with mixed-integer logistic network design problems ("supply chain models") with dominant special structure, such as embedded networks.

During branch-and-bound enumeration of large, difficult problems, we have found that it is preferable to make a considerable investment in computation before committing to solve another linear program. In this setting, VIP penalties require an inconsequential amount of additional code and computation.

There are two sources of VIP penalties: those based on existing model constraints, and those derived from valid integer polyhedral constraints.

We constantly monitor restricted simplicial problem structure during enumeration to identify and to isolate candidate rows for constraint-based branching. Thus, identification of existing restricted source constraints (e.g., a 1-cover with two or more binary variables with fractional values) is trivial.

We also seek violated integer polyhedral cuts during enumeration (e.g., [2,5]). Most such candidate cuts are of a form useful for VIP penalties. The VIP penalty for a violated cut offers a lower bound on the value of the cut as a simplicial restriction.

VIP penalties are additive across independent simplicial components that are often created by a sequence of restrictions within branch and bound. We use breadth-first search to label and isolate disjoint simplicial components; VIP penalties are easily computed and accumulated across these. Our set partitioning problems commonly break into 50–100 disjoint components during nontrivial enumerations, while supply chain problems seldom exhibit more than five.

VIP penalties are computed in polynomial time for k-pack, k-cover, k-partition and k-VUB constraints, with computational effort proportional to $k \log k$. Our experiments suggest that limiting k to 1 minimizes computational overhead with little loss in effectiveness.

So, the good news is that VIP penalties are easy to compute. The bad news is that we never know the extent to which they will help.

With almost all models, VIP penalties improve the lower bound by a nonzero amount. However, most improvements are small – a fraction of a percent reduction in the integrality gap is typical. However, if we eliminate trivial enumeration problems, say, those that render the desired integrality gap within 20 branches, the remaining models are more likely to be improved by VIP.

An example of an outlier we came across is a set partition with 1,200 rows, 36,400 binary columns, and an average of 50 nonzero coefficients per column. Here, a stubborn residual integrality gap of 18% is suddenly reduced to zero by the sum of VIP penalties dominated by two of 23 extant disjoint simplicial components. The consequent fathom soon resolves the problem. The key source rows are a pair of simple 1-partitions in the signal restriction, with the two dominating penalties each deriving from the 1-cover case (5). Repeating the exercise many times, varying enumeration strategy to suppress various accompanying features (including, in particular, integer polyhedral cuts), we find a number of similar "VIP events", but no discernable explanatory pattern. Across these experiments, VIP frequently plays a key role in resolving an otherwise interminable enumeration.

Not surprisingly, VIP penalties are more valuable if not competing with aggressive generation of polyhedral cuts. Comparatively, the cuts are more powerful and reliable, but much more expensive to find and apply. We have retained both as options, suggesting them as defaults when a model is dominated by set-partitioning constraints.

Another significant “VIP event” arises in the Benders master problem of a decomposed supply-chain model. The original master problem has 513 rows and 32,110 binary columns, but a single 1-cover (5) isolated in a restriction reduces an integrality gap of 8% to less than 1%. At the point that this restriction is generated, there are only two disjoint components, and no discovered polyhedral cuts. Without VIP penalties, the time to resolve this master problem increases by an order of magnitude, and then another decomposition iteration of comparable difficulty is required.

Sometimes, you really want to close the integrality gap completely, rather than just to some moderate tolerance. In these cases, VIP penalties frequently appear to significantly reduce the exhaustive enumeration effort.

We have adopted VIP penalties as a default enumeration feature in our ILP applications.

Acknowledgements

We thank Naval Postgraduate School Professors Kevin Wood, Alexandra Newman, and Alan Washburn for their reviews.

References

[1] G.G. Brown, M.P. Olson, Dynamic factorization in large-scale optimization, Math. Programming 64 (1994) 17–51.
[2] H. Crowder, E.L. Johnson, M. Padberg, Solving large scale zero-one linear programming problems, Oper. Res. 31 (1983) 803–834.
[3] G.B. Dantzig, R.M. Van Slyke, Generalized upper bounding techniques, J. Comput. System Sci. 1 (1967) 213–226.
[4] R.S. Garfinkel, G.L. Nemhauser, Integer Programming, Wiley, New York, 1972.
[5] K.L. Hoffman, M. Padberg, Solving airline crew scheduling problems by branch-and-cut, Management Sci. 39 (1993) 657–682.
[6] G.L. Nemhauser, L.A. Wolsey, Integer and combinatorial optimization, Wiley, New York, 1988.
[7] R.G. Parker, R.L. Rardin, Discrete Optimization, Academic Press, New York, 1988.
[8] L. Schrage, Implicit representation of variable upper bounds in linear programming, Math. Programming 4 (1975) 118–132.
[9] J.A. Tomlin, Branch and bound methods for integer and non-convex programming, in: J. Abadie (Ed.), Integer and Nonlinear Programming, American Elsevier, 1970, pp. 437–450.
[10] J.A. Tomlin, An improved branch and bound method for integer programming, Oper. Res. 19 (1971) 1070–1075.