DIMACS Reconnect Conference on MIP

Preprocessing and Probing for Integer Programs
(part I)

Jeff Linderoth

June 23 2004

Outline

- Preprocessing
- Probing

Preprocessing and Probing

- Effective preprocessing of MIP can pay large dividends
  - ◇ Some preprocessing techniques (like redundant constraint identification and variable fixing) are also used to help solve linear programs
- Most of this taken from the "Preprocessing and Probing" paper by Martin Savelsbergh.
- Another good paper, from which much of MINTO's "advanced" and CPLEX's preprocessing is taken is Conflict Graphs in Solving Integer Programming Problems, A. Atamtürk, G.L. Nemhauser and M.W.P. Savelsbergh, European Journal of Operational Research 121, 40-55, 2000
- $l_j \leq x_j \leq u_j$

Implied Bounds

- The most basic type of preprocessing is calculating *implied bounds*.
- Let (π, π_0) be a valid inequality.
  - ◇ In "IP Speak" this means that there might be a row in the formulation of the form $\pi^T x \leq \pi_0$
- If $\pi_1 > 0$, then

$$x_1 \leq (\pi_0 - \sum_{j:\pi_j>0} \pi_j l_j - \sum_{j:\pi_j<0} \pi_j u_j)/\pi_1$$

  - ◇ Note slight abuse of notation.
  - ◇ Why?

$$\begin{aligned}
\pi_1 x_1 &= \pi_0 - \sum_{j \in N \setminus \{1\} : \pi_j > 0} \pi_j x_j - \sum_{j \in N \setminus \{1\} : \pi_j < 0} \pi_j x_j \\
&\leq \pi_0 - \sum_{j \in N \setminus \{1\} : \pi_j > 0} \pi_j l_j - \sum_{j \in N \setminus \{1\} : \pi_j > 0} \pi_j u_j .
\end{aligned}$$

Similarly

- If $\pi_1 < 0$, then

$$x_1 \geq (\pi_0 - \sum_{j:\pi_j>0} \pi_j l_j - \sum_{j:\pi_j<0} \pi_j u_j)/\pi_1$$

- Lots and lots of preprocessing techniques are based on these simple bounding arguments

Basic Preprocessing

- Again, let (π, π_0) be any valid inequality for S.
- The constraint $\pi x \leq \pi_0$ is redundant if

$$\sum_{j:\pi_j>0} \pi_j u_j + \sum_{j:\pi_j<0} \pi_j l_j \leq \pi_0.$$

- S is empty (IP is infeasible) if

$$\sum_{j:\pi_j>0} \pi_j l_j + \sum_{j:\pi_j<0} \pi_j u_j > \pi_0.$$

- For any IP of the form $\max\{c^T x | Ax \leq b, l \leq x \leq b\}, x \in \mathbf{Z}^n$,
  - If $a_{ij} \geq 0 \forall i \in [1..m]$ and $c_j < 0$, then $x_j = l_j$ in any optimal solution.
  - If $a_{ij} \leq 0 \forall i \in [1..m]$ and $c_j > 0$, then $x_j = u_j$ in any

optimal solution.

Probing for Integer Programs

- It is also possible in many cases to fix variables or generate new valid inequalities based on logical implications.
- Consider (π, π_0), a valid inequality for 0-1 integer program.
- If $\pi_k > 0$ and $\pi_k + \sum_{j:\pi_j<0} \pi_j > \pi_0$, then we can fix x_k to zero.
- Similarly, if $\pi_k < 0$ and $\sum_{j:\pi_j<0, j\neq k} \pi_j > \pi_0$, then we can fix x_k to one.
- Example: Generating logical inequalities

Improving Coefficients

- Suppose again that (π, π_0) is a valid inequality for a 0-1 integer program.
- Suppose that $\pi_k > 0$ and $\sum_{j:\pi_j>0, j\neq k} \pi_j < \pi_0$.
- If $\pi_k > \pi_0 - \sum_{j:\pi_j>0, j\neq k} \pi_j$, then we can set
  - $\pi_k \leftarrow \pi_k - (\pi_0 - \sum_{j:\pi_j>0, j\neq k} \pi_j)$, and
  - $\pi_0 \leftarrow \sum_{j:\pi_j>0, j\neq k} \pi_j)$.
- Similarly, suppose that $\pi_k < 0$ and $\pi_k + \sum_{j:\pi_j>0, j\neq k} \pi_j < \pi_0$.
- Then we can again set $\pi_k \leftarrow \pi_k - (\pi_0 - \pi_j - \sum_{j:\pi_j>0, j\neq k} \pi_j)$

Bound Improvement by Reduced Cost

- Consider an integer program $\max\{cx \mid Ax \leq b, 0 \leq x \leq u\}$
- Suppose the linear programming relaxation has been solved to optimality and row zero of the tableau looks like

$$z = \bar{a}_{00} + \sum_{j \in NB_1} \bar{a}_{0j} x_j + \sum_{j \in NB_2} \bar{a}_{0j}(x_j - u_j)$$

  where NB_1 are the nonbasic variables at 0 and NB_2 are the nonbasic variables at their upper bounds u_j.
- In addition, suppose that a lower bound $\underline{z}$ on the optimal solution value for IP is known.

- Then in any optimal solution

$$\begin{aligned} x_j &\leq \left\lfloor \frac{\bar{a}_{00} - \underline{z}}{-\bar{a}_{0j}} \right\rfloor \text{ for } j \in NB_1, \text{ and} \\ x_j &\geq u_j - \left\lceil \frac{\bar{a}_{00} - \underline{z}}{\bar{a}_{0j}} \right\rceil \text{ for } j \in NB_2. \end{aligned}$$

Preprocessing and Probing in Branch and Bound

- In practice, these rules are applied iteratively until none applies.
- Applying one of the rules may cause a new rule to apply.
- Bound improvement by reduced cost can be reapplied whenever a new bound is computed.
- Furthermore, all rules can be reapplied after branching.
- These techniques can make a very big difference.
- **How big, you ask?**

"Objective Function" Preprocessing

max

$$3x_1 + 2x_2 - 5x_3$$

subject to

$$\begin{aligned} x_1 + x_2 + x_3 &\leq 1 \\ -x_1 + 2x_3 &\leq 4 \\ x &\in \mathbb{B}^3 \end{aligned}$$

min

$$\pi_1 + 4\pi_2 + \mu_1 + \mu_2 + \mu_3$$

subject to

$$\begin{aligned} \pi_1 - \pi_2 + \mu_1 &\geq 3 \\ \pi_1 + \mu_2 &\geq 2 \\ \pi_1 + 2\pi_2 + \mu_3 &\geq -5 \\ \pi &\in \Re^2_+ \\ \mu &\in \Re^3_+ \end{aligned}$$

The Conflict Graph

- This "triggering" can be conveniently captured in the notion of the "conflict graph".
- Create a graph $G = ((B \cup B^C), E)$, where E is the set of variables or their complements that can not both simulaneously be one.
  - (i.e., you have found some implications of the form $x_i = 1 \Rightarrow x_j = 0$)
- An example