Lookahead Branching for MIP

Wasu Glankwamdee and Jeff Linderoth

Today's Outline

- Strong Branching
- Algorithm & Branching Rules
- Algorithmic Enhancements & Speed-Up
- Computational Results
- Conclusions & Future Research

MIP Formulation

Maximize

$$z_{MIP} = \sum_{j \in I} c_j x_j + \sum_{j \in C} c_j x_j$$

subject to

$$\begin{aligned}
\sum_{j \in I} a_{ij} x_j + \sum_{j \in C} a_{ij} x_j &\leq b_i & i \in M, & \quad (1) \\
l_j \leq x_j &\leq u_j & j \in N, & \quad (2) \\
x_j &\in \mathcal{Z}_+ & j \in I, & \quad (3) \\
x_j &\in \mathcal{R}_+ & j \in C. & \quad (4)
\end{aligned}$$

Branching is Important

- Effective branching is more important near the top of the tree.
- We might want to evaluate more candidates near the top of the tree.
- More candidates almost always result in smaller trees, but the expense eventually causes an increase in running time.

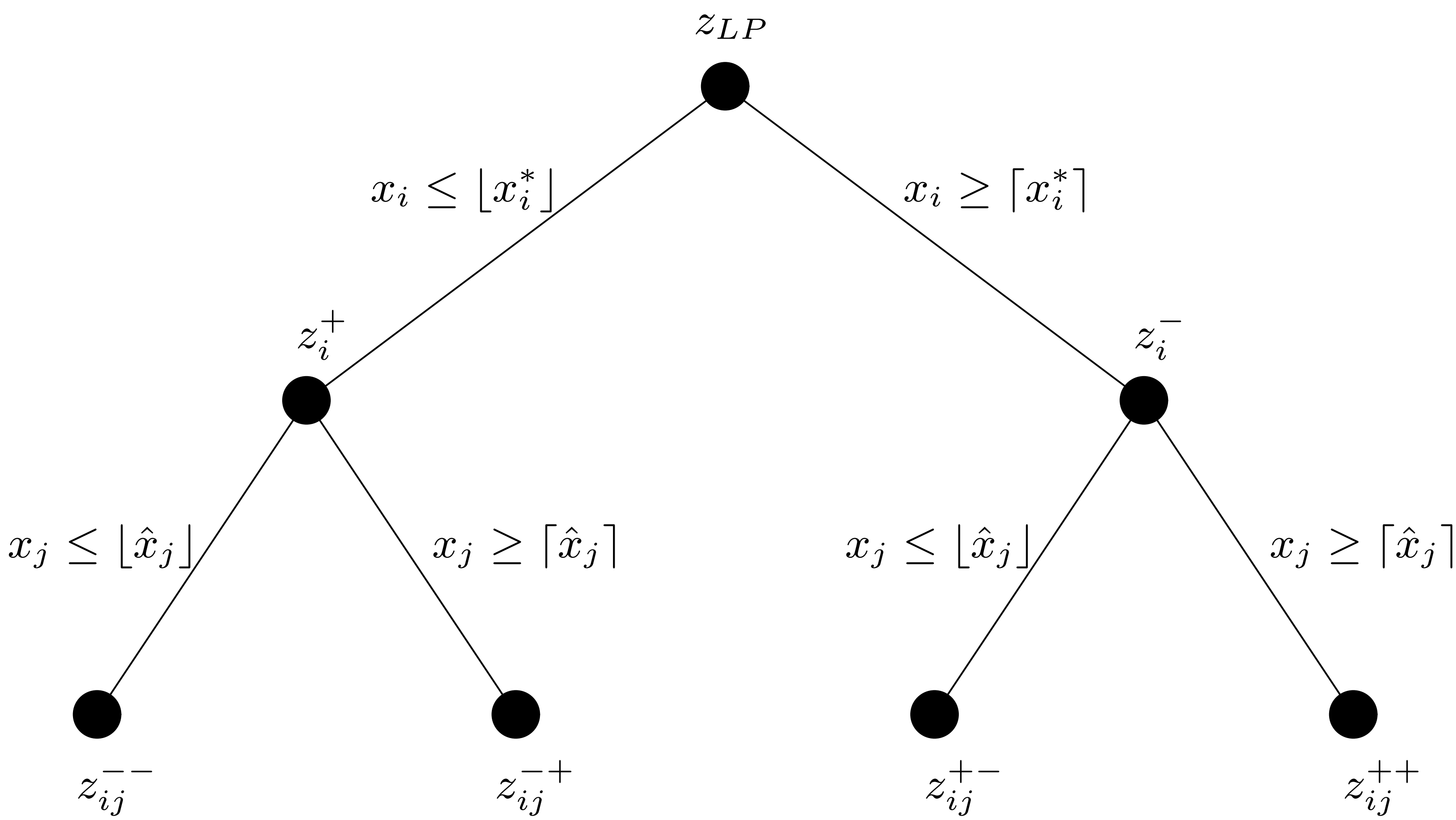
Strong Branching

- Select a set C of basic fractional variables to branch up and down, and perform a specific number of dual simplex pivots on each variable in this set.
- How do we choose the set C?
  - ◇ x_j for which the values are furthest from being an integer. For 0-1 variable, this means those whose values are closest to 0.5.
  - ◇ x_j for which the values are sufficiently fractional and the objective function coefficients are the largest.
  - ◇ x_j for which the pseudocosts are the largest.

Motivation

- Can we do better by taking into account the branching information two-level deeper than the current local node?
- Can better branching decisions be made?
- “Ramp-Up”

Two-Levels Deep Search Tree



Definitions

- $\mathcal{G}_i^- = \{j \in \mathcal{F}_i^- | \rho_{ij}^{--} = 0, \rho_{ij}^{-+} = 0\}$.
- $\mathcal{G}_i^+ = \{j \in \mathcal{F}_i^+ | \rho_{ij}^{+-} = 0, \rho_{ij}^{++} = 0\}$.
  - The sets of indices of fractional variables in the corresponding feasible LP relaxations two-levels deep.
- $\mathcal{W}(a, b) = \{\alpha_1 \min(a, b) + \alpha_2 \max(a, b)\}$.
  - Weighting function.
- $D_{ij}^{s_1 s_2} = z_{LP} - z_{ij}^{s_1 s_2}$, where$s_1, s_2 \in -, +$.
  - The degradation in LP relaxation value two-levels deep.

Branching Rules

- Rule 1: Maximize Best Degradation

$$i^* = \arg\max_{i \in \mathcal{F}} \left\{ \max_{j \in \mathcal{G}_i^-} \{\mathcal{W}(D_{ij}^{--}, D_{ij}^{-+})\} + \max_{j \in \mathcal{G}_i^+} \{\mathcal{W}(D_{ij}^{+-}, D_{ij}^{++})\} \right\}.$$

- Rule 2: Maximize Sum of Degradation

$$i^* = \arg\max_{i \in \mathcal{F}} \left\{ \frac{1}{|\mathcal{G}_i^-|} \sum_{j \in \mathcal{G}_i^-} \mathcal{W}(D_{ij}^{--}, D_{ij}^{-+}) + \frac{1}{|\mathcal{G}_i^+|} \sum_{j \in \mathcal{G}_i^+} \mathcal{W}(D_{ij}^{+-}, D_{ij}^{++}) \right\}.$$

Branching Rules

- Rule 3: Maximize Number of Infeasibility

$$i^* = \arg\max_{i \in \mathcal{F}} \eta_i.$$

- Rule 4: Maximize Degradation and Number of Infeasibility

$$i^* = \arg\max_{i \in \mathcal{F}} \left\{ \max_{j \in \mathcal{F}_i^-} \{\mathcal{W}(D_{ij}^{--}, D_{ij}^{-+})\} + \max_{j \in \mathcal{F}_i^+} \{\mathcal{W}(D_{ij}^{+-}, D_{ij}^{++})\} - \beta\eta_i \right\}.$$

Variables' Bound Fixing

Derivation	Implication
$\xi_i^- = 1$	$x_i \geq \lceil x_i^* \rceil$
$\xi_i^+ = 1$	$x_i \leq \lfloor x_i^* \rfloor$
$\rho_{ij}^{--} = 1$ and $\rho_{ij}^{-+} = 1$	$x_i \geq \lceil x_i^* \rceil$
$\rho_{ij}^{+-} = 1$ and $\rho_{ij}^{++} = 1$	$x_i \leq \lfloor x_i^* \rfloor$

Clique Inequalities

Derivation	Implication
$\rho_{ij}^{--} = 0$	$(1 - x_i) + (1 - x_j) \leq 1$
$\rho_{ij}^{+-} = 0$	$(1 - x_i) + x_j \leq 1$
$\rho_{ij}^{+-} = 0$	$x_i + (1 - x_j) \leq 1$
$\rho_{ij}^{++} = 0$	$x_i + x_j \leq 1$

Computational Results

Branching Rule	Avg. Ranking	
	w/ Fix&Cut	w/o Fix&Cut
One-Level	4.40	2.85
Rule 1	2.46	3.60
Rule 2	2.13	2.43
Rule 3	3.13	3.05
Rule 4	2.88	4.08

Table 1: Summary of Experiments

Computational Results

Branching Rule	# Evaluated Nodes	
	w/ Fix&Cut	w/o Fix&Cut
MINTO Default	16974	16974
One-Level	8471	8471
Rule 1	1319	8946
Rule 2	946	8004
Rule 3	1571	8145
Rule 4	1191	8832

Table 2: Average Number of Evaluated Nodes in Solved Instances

Computational Results

Branching Rule	Avg. Integrality Gap	
	w/ Fix&Cut	w/o Fix&Cut
One-Level	45.36	45.36
Rule 1	9.41	47.70
Rule 2	9.22	43.02
Rule 3	11.29	45.98
Rule 4	9.60	48.80

Table 3: Average Integrality Gap in Unsolved Instances

Speed-Up

- Limit the number of simplex iterations on all fractional variables at two-levels deep nodes.
- Limit the number of fractional variables on which to perform simplex iteration both at one-level and two-levels deep node, i.e. reduce the size of the candidate branching set.

Computational Results

Branching Rule	# Evaluated Nodes		
	(limit iter.)	(limit frac.)	(w/o limit)
One-Level	8608	8623	8471
Rule 1	9106	10422	8946
Rule 2	8321	10272	8004
Rule 3	8140	8337	8145
Rule 4	8668	10386	8832

Table 4: Average Number of Evaluated Nodes in Solved Instances

Computational Results

Branching Rule	Avg. Integrality Gap		
	(limit iter.)	(limit frac.)	(w/o limit)
One-Level	38.99	31.08	45.36
Rule 1	42.57	41.54	47.70
Rule 2	40.57	41.56	43.02
Rule 3	40.04	31.94	45.98
Rule 4	42.49	41.54	48.80

Table 5: Average Integrality Gap in Unsolved Instances

Computational Results

Branching Rule	Avg. Integrality Gap		
	(limit iter.)	(limit frac.)	(w/o limit)
One-Level	59.73	59.70	57.28
Rule 1	60.05	62.99	58.18
Rule 2	59.96	62.99	53.99
Rule 3	60.67	59.97	57.56
Rule 4	59.99	62.99	59.79

Table 6: Average Integrality Gap When the Same Number of Nodes Are Solved

Conclusions

- There exists significantly important branching information at two-levels deep.
- The branching rules often reduce the size of the search tree in comparison to “full” strong branching, and to branching rules implemented in commercial solvers.
- Tighter representation of MIP and an even smaller branch and bound tree are achieved by incorporating preprocessing and probing techniques.
- Similar branching decision can still be made, but with less computational effort, by limiting number of simplex iterations or the number of fractional variables.

Future Research

- Can we develop other useful branching rules based on measuring the degradation in LP relaxation value two-levels deep? We are particularly interested in methods based on multiobjective optimization, extending our branching rule 4.
- Can we derive implication inequalities for general integer variables at two-levels deep?
- Can we speed up the two-levels deep branching algorithm even more by imposing the limitation on both the number of simplex iterations and the number of fractional variables?
- Can the ideas presented here be incorporated into practical methods for integer programming to solve larger problems?