Primal heuristics in MIPs

Presented by:

Kumar Abhishek (Lehigh University)

Ashutosh Mahajan (Lehigh University)

Outline of Talk

- Introduction: MIP and Primal Heuristic
- Description of two primal heuristics
- Implementation
- Proposed Work

Primal Heuristics in MIP (1)

- A typical MIP solver uses Branch and Bound (and Cuts)
- A Linear Program is solved at each node
- There are 3 possibilities
  1. Solution of LP is feasible to the original problem
  2. Solution of the LP is non integral and hence infeasible for the original problem
  3. LP is infeasible
- If solution is non-integral, there are two possibilities
  1. $z_{LP} \geq$ best solution $\Rightarrow$ Fathom
  2. $z_{LP} \leq$ best solution $\Rightarrow$ sub-branching

Hence a good incumbent is important.

Primal Heuristics in MIP (2)

A good incumbent may be achieved by:

- Fixing variables and diving
- Searching around the current LP solution for an integral solution
- If an IP solution is found, searching for a better solution in the neighbourhood.

None of the approaches guarantee a good solution.

None of them guarantee good speed

Good heuristics have been limited to specific problem structures

Our Objective

To implement generalized primal heuristics for an MIP solver

Proposed Heuristics

1. Local Branching (Fischetti and Lodi - 2002)
2. RINS (Danna, Rothberg and Pape - 2004)

MIP solvers

1. MINTO
2. SYMPHONY

LOCAL BRANCHING

- A form of soft-fixing: give solver some freedom to fix variables e.g.

$$\Delta(x, \bar{x}) = \sum_{j \in \bar{S}} (1 - x_j) + \sum_{j \in B \setminus \bar{S}} x_j \leq k$$

  where,

$$\bar{S} = \{j \in B : \bar{x_j} = 1\}$$

- Branching decision

$$\Delta(x, \bar{x}^1) \leq k$$

$$\Delta(x, \bar{x}^1) \geq k + 1$$

- Iterative procedure
- Additionally time-bounds and diversification could be used.

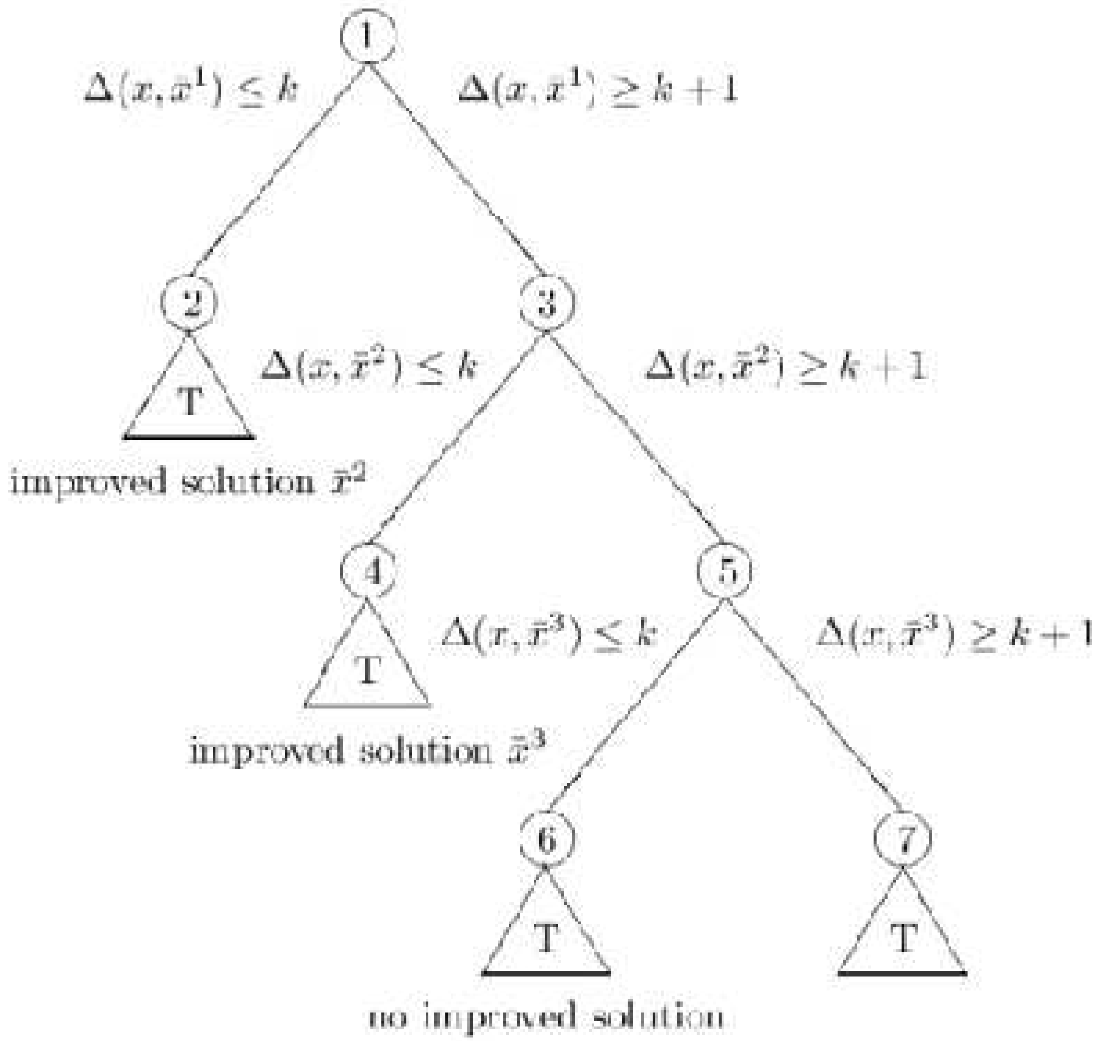


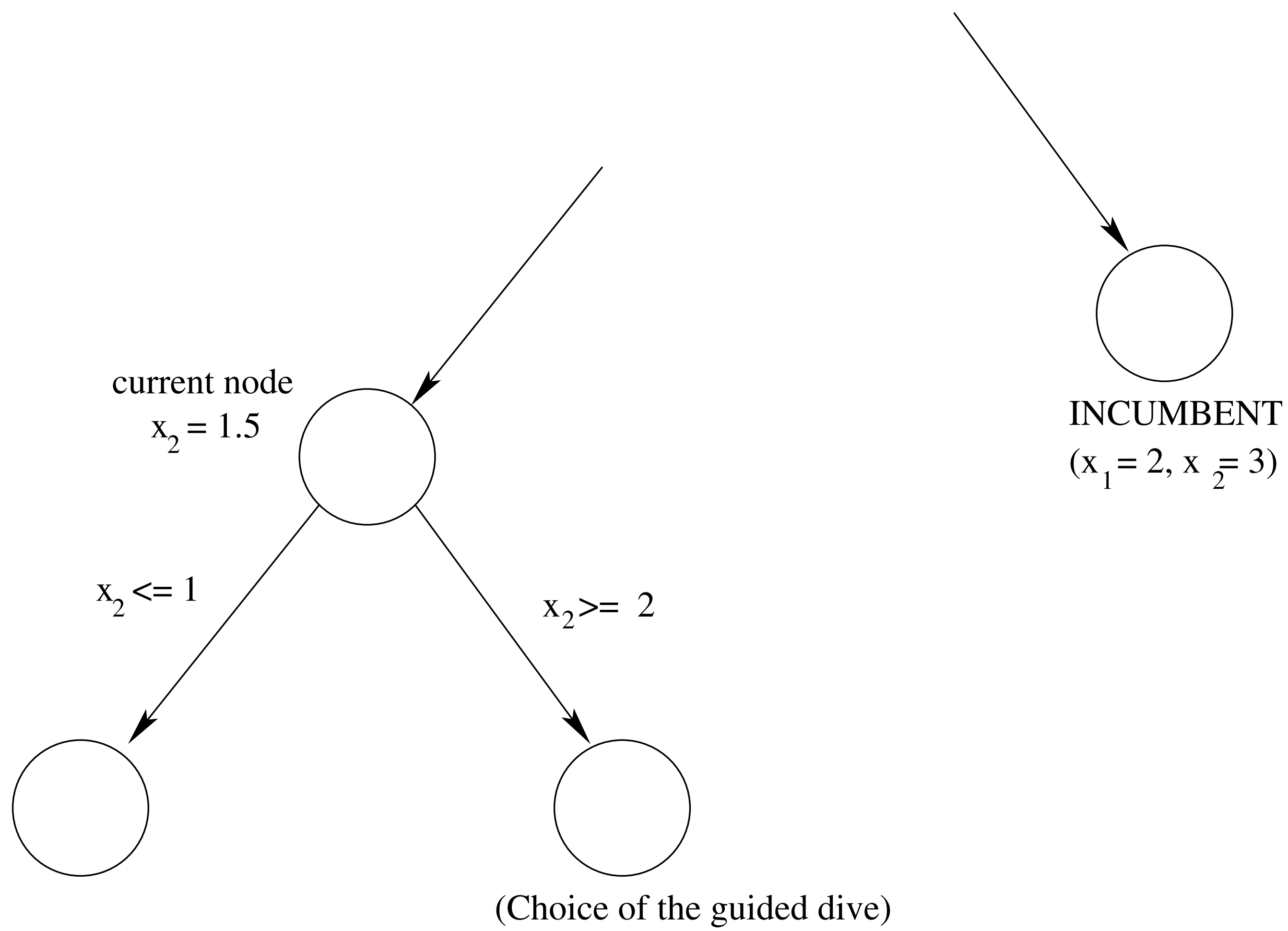
Figure 1: The basic local branching scheme.

Relaxation Induced Neighborhood Search (RINS)

- Assumes that an incumbent exists.
- Intuitively, a good solution should be around the LP Optimal. Also, it should have some closeness to the incumbent.
- Fix those variables which have the same value in the incumbent and the LP optimal.
- Form a new MIP after fixing these variables.
- Solve this simpler MIP.
- Optionally, put a limit on the number of nodes to be searched in the sub-MIP.

Guided Dives

- Extends the idea of RINS while deciding which branch to choose first.
- Chooses that branch which allows the branching variable to take the value it has in the current incumbent.
- If this branch does not yield a solution then go to the other branch.



EXAMPLE OF GUIDED DIVE

Implementation

- RINS Requires a call to the MIP solver to solve a smaller sized MIP.
- In MINTO, this can be achieved through recursion.
- A new problem will be formulated using the APPL functions.
- The original formulation will be kept. The branching constraints will be dropped.
- Solve this simpler MIP.
- Return the result back to the parent.
- Based on the results obtained from the child, update the parent.
- In case of SYMPHONY, recursion is now possible.

Implementation-II

- Local Branching:
  - No new MIP instance required.
  - Add soft bound constraints at a new incumbent and resolve.
  - May need to backtrack by removing the bound constraints.
  - Other issues:
    * Finding suitable parameters like maximum sub-problem size, branching parameters, time-limits etc.