MIP Heuristics
Motivation for Heuristics

Why not wait for branching?

• **Produce feasible solutions as quickly as possible**
  – Often satisfies user demands
  – Avoid exploring unproductive sub-trees
  – Better reduced-cost fixing

• **Avoid “tree pollution”**
  – Good fixings in a heuristic are often not good branches

• **Increase diversity of search**
  – Strategies in heuristic may differ from strategies in branching
Two Traditional Classes of Heuristics

• Plunging heuristics:
  – Maintain linear feasibility
  – Try to achieve integer feasibility

• Local improvement heuristics:
  – Maintain integer feasibility
  – Try to achieve linear feasibility
Plunging Heuristic Structure

• Fix a set of integer infeasible variables
  – Usually by rounding
• Perform bound strengthening to propagate implications
• Solve LP relaxation
• Repeat
Bound Strengthening
Propagate new bounds through inequalities

- Given a constraint:
  - $\sum a_j x_j \leq b$
  - Split equalities into a pair of inequalities
- Consider a single $x_k$:
  - $a_k x_k + \inf (\sum_{j \neq k} a_j x_j) \leq \sum a_j x_j \leq b$
  - $x_k \leq (b - \inf (\sum_{j \neq k} a_j x_j)) / a_k$
    - Assuming $a_k \geq 0$
- Change in variable bound can produce changes in other bounds
Bound Strengthening Example

• $x + 2y + 3z \leq 3$
  ▪ all variables binary
  ▪ $x=1$
• $3 z \leq 3 - \inf (x + 2y) = 3 - 1 = 2$
• $z \leq 2/3$
Plunging Details

Important details

• How many variables to fix per round:
  – All of them?
    • Inexpensive; no need to solve LP relaxations
    • But ‘flying blind’ after a few fixings
      – Bound strengthening helps
  – A few?
    • More expensive
    • LP relaxation can guide later choices
      – (variable values, reduced costs, etc.)

• In what order are variables fixed?
  – Variations useful for diversification
Local Improvement Heuristics
High-level structure

• Choose integer values for all integer variables
  – Produces linear infeasibility

• Iterate over integer variables:
  – Does adding/subtracting 1 reduce linear infeasibility?

• Infeasibility metrics:
  – Primary: number of violated constraints
  – Secondary: |b-Ax|
Local Improvement Details

• What initial values to assign to integer variables?
  – Rounded relaxation values
  – 0

• Move acceptance criteria?
  – Greedy

• What to do when local improvement gets stuck?
  – Reverse infeasibility metrics
Sub-MIP As A Paradigm

• Key recent insight for heuristics:
  – Can use MIP solver recursively as a heuristic
  – Solve a related model:
    • Hopefully smaller and simpler
  – Examples:
    • Local cuts [Applegate, Bixby, Chvátal & Cook, 2001]
    • Local branching [Fischetti & Lodi, 2003]
    • RINS [Danna, Rothberg, Le Pape, 2005]
    • Solution polishing [Rothberg, 2007]
Local Branching
Viewed as an Exact Method

• Local Branching [Fischetti and Lodi, 2002]
  – Assume an integer feasible solution $x^*$ is known. Label this solution the incumbent.
  – Step1:
    a. Add the “local branching” constraint $|x - x^*| \leq k$
    b. Solve this MIP
    c. Replace the added constraint by $|x - x^*| \geq k + 1$
    d. If a new incumbent $x^{**}$ was found in (b) replace $x^*$ by $x^{**}$ and return to (a).
  – Step2: Solve the resulting MIP.
Local Branching
Viewed as a Heuristic

• Constrain sub-MIP to explore a small neighborhood of incumbent $x^*$
  – $|x - x^*| \leq k$
  – $k$ chosen to be ~20
  – Impose node limit on sub-MIP search
  – $k$ can be adjusted dynamically

• Apply whenever a new incumbent is found
  – Including those found by local branching

• A succession of improving, neighboring solutions
RINS

• RINS [Danna, Rothberg, Le Pape, 2005]
• Relaxation Induced Neighborhood Search
  – Given two “solutions”:
    • $x^*$: any integer feasible solution (not optimal)
    • $x^R$: optimal relaxation solution (not integer feasible)
      – Fix variables that agree
      – Solve the result as a MIP
• Possibly requiring early termination
• Extremely effective heuristic
  – Often finds solutions that no other technique finds
RINS
Implementation

• Dynamically adjust future fixing fraction based on result of sub-MIP solution:
  – Sub-MIP finds seed solution:
    • Sub-MIP is too easy - fix fewer variables next time
  – Sub-MIP does not find seed solution:
    • Sub-MIP is too hard - fix more variables next time
  – Sub-MIP finds better solution:
    • Sub-MIP is just right
RINS
Implementation – “Goldilocks Method”

![RINS Chart]

- ljb12
- sp97ar
- rococoB12-100000
RINS

Why is it so Effective?

• MIP models often involve a hierarchy of decisions
  – Some much more important than others
• Fixing variables doesn’t just make the problem smaller
  – Often changes the nature of the problem
    • Extreme case:
      – Problem decomposes into multiple, simple problems
    • More general case:
      – Resolving few key decisions can have a dramatic effect
  – Strategies that worked well for the whole problem
    may not work well for RINS sub-MIP
    • More effective to treat it as a brand new MIP
Solution Polishing
An Evolutionary Algorithm

• Solution polishing [Rothberg, 2007]
• Three crucial components:
  – Selection:
  • Choose a pair of candidate solutions
  • More fit candidates more likely to be chosen
  – Combination:
  • Combine the chosen pair to produce an offspring
  – Mutation:
  • Allow the offspring to vary from the parents in some (random) way
Solution Polishing
The Population

• A single solution pool
  – Contains 40 best solutions
    • Ties are broken on age
      – Younger solutions push out older ones

• New solutions added immediately
  – No notion of generations
    • Mutation and combination quite expensive
    • Need to integrate new solutions quickly

• Solutions from regular MIP search also added to candidate pool
  – Tree search and evolutionary algorithm cooperate
Solution Polishing

Mutation

• Apply a random mask vector:

Seed solution:  
1 0 1 0 0 1 1

Random mask:  

Mutation:  
? 0 ? 0 0 ? 1

• Solve truncated sub-MIP:
  • Only masked values allowed to differ from seed solution
  • Use Goldilocks method to determine how many to fix
Solution Polishing
Combination

• Only variables whose values differ in parents are allowed to vary in offspring

```
Parent 1:  1 0 1 0 0 1 1
Parent 2:  1 1 1 1 0 1 0
           ↓
Offspring: 1 ? 1 ? 0 1 ?
```

• Solve truncated sub-MIP
• Occasionally combine all solutions
Solution Polishing

Selection

• Selection method empirically not very important
  – Modest population size

• Simplest strategy worked well:
  – Pick a random parent from solution pool
  – Pick a random pair from among those with better objectives than the first
Solution Polishing
Putting it all Together

Solution pool

Evolutionary heuristic
- Mutation
- Combination
Rethinking MIP Tree Search
Sub-MIP As A Paradigm

• Key recent insight for heuristics:
  – Can use MIP solver recursively as a heuristic
  – Solve a related model:
    • Hopefully smaller and simpler
  – Examples:
    • Local cuts [Applegate, Bixby, Chvátal & Cook, 2001]
    • Local branching [Fischetti & Lodi, 2003]
    • RINS [Danna, Rothberg, Le Pape, 2005]
    • Solution polishing [Rothberg, 2007]
RINS
Why is it so Effective?

• MIP models often involve a hierarchy of decisions
  – Some much more important than others
• Fixing variables doesn’t just make the problem smaller
  – Often changes the nature of the problem
    • Extreme case:
      – Problem decomposes into multiple, simple problems
    • More general case:
      – Resolving few key decisions can have a dramatic effect
  – Strategies that worked well for the whole problem may not work well for RINS sub-MIP
    • More effective to treat it as a brand new MIP
Tree-of-Trees

• Gurobi MIP search tree manager built to handle multiple related trees
  – Can transform any node into the root node of a new tree

• Maintains a pool of nodes from all trees
  – No need to dedicate the search to a single sub-tree
Tree-of-Trees
Tree-of-Trees

• Each tree has its own relaxation and its own strategies...
  – Presolved model for each subtree
  – Cuts specific to that subtree
  – Pseudo-costs for that subtree only
  – Symmetry detection on that submodel
  – Etc.

• Captures structure that is often not visible in the original model
Summary of Heuristics

• 5 heuristics prior to solving root LP
  – 5 different variable orders, fix variables in this order

• 15 heuristics within tree (9 primary, several variations)
  – RINS, rounding, fix and dive (LP), fix and dive (Presolve), Lagrangian approach, pseudo costs, Hail Mary (set objective to 0)

• 3 solution improvement heuristics
  – Applied whenever a new integer feasible is found
Performance
Gurobi Optimizer version 2.0.0

Set parameter heuristics to value 0

Read MPS format model from file ns1671066.mps.bz2

ns167106: 316 Rows, 2840 Columns, 31418 NonZeros
Presolved: 315 Rows, 1819 Columns, 19336 Nonzeros

Root relaxation: objective 7.634608e+00, 241 iterations, 0.01 seconds

<table>
<thead>
<tr>
<th>Nodes</th>
<th>Current Node</th>
<th>Objective Bounds</th>
<th>Work</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Expl Unexpl</td>
<td>Obj Depth IntInf</td>
<td>Incumbent BestBd Gap</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>7.6346 0 20</td>
<td>7.6346 - - 0s</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>7.6346 0 34</td>
<td>7.6346 - - 0s</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>7.6346 0 2</td>
<td>7.6346 - - 0s</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>7.6346 0 25</td>
<td>7.6346 - - 0s</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>7.6346 0 6</td>
<td>7.6346 - - 0s</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>7.6346 0 6</td>
<td>7.6346 - - 0s</td>
</tr>
<tr>
<td>* 1998</td>
<td>1716</td>
<td>9.1334 7.6346 16.4% 25.2 1s</td>
<td></td>
</tr>
<tr>
<td>* 2002</td>
<td>1710</td>
<td>9.1031 7.6346 16.1% 25.1 1s</td>
<td></td>
</tr>
<tr>
<td>* 2172</td>
<td>1359</td>
<td>8.3611 7.6346 8.69% 23.9 1s</td>
<td></td>
</tr>
<tr>
<td>* 2177</td>
<td>1358</td>
<td>8.3608 7.6346 8.69% 23.8 1s</td>
<td></td>
</tr>
<tr>
<td>4467</td>
<td>2736</td>
<td>8.3508 7.6346 8.69% 23.0 5s</td>
<td></td>
</tr>
<tr>
<td>* 5695</td>
<td>3015</td>
<td>8.3453 7.6346 8.52% 20.7 5s</td>
<td></td>
</tr>
<tr>
<td>23241</td>
<td>15991</td>
<td>8.3453 7.6346 8.52% 13.7 10s</td>
<td></td>
</tr>
<tr>
<td>47601</td>
<td>35137</td>
<td>8.3453 7.6346 8.52% 11.1 15s</td>
<td></td>
</tr>
<tr>
<td>*55945</td>
<td>37046</td>
<td>8.2735 7.6346 7.72% 10.6 16s</td>
<td></td>
</tr>
<tr>
<td>*70873</td>
<td>48462</td>
<td>8.2724 7.6346 7.71% 10.1 19s</td>
<td></td>
</tr>
<tr>
<td>*71445</td>
<td>48891</td>
<td>8.2715 7.6346 7.70% 10.1 19s</td>
<td></td>
</tr>
<tr>
<td>72961</td>
<td>50242</td>
<td>8.2715 7.6346 7.70% 10.0 20s</td>
<td></td>
</tr>
<tr>
<td>91653</td>
<td>64329</td>
<td>8.2715 7.6346 7.70% 10.4 25s</td>
<td></td>
</tr>
<tr>
<td>*79720</td>
<td>47515</td>
<td>8.2715 7.6346 7.70% 10.4 25s</td>
<td></td>
</tr>
<tr>
<td>111094</td>
<td>57352</td>
<td>8.2715 7.6346 7.70% 10.4 25s</td>
<td></td>
</tr>
<tr>
<td>*125331</td>
<td>50985</td>
<td>8.2715 7.6346 7.70% 10.4 25s</td>
<td></td>
</tr>
<tr>
<td>133884</td>
<td>65918</td>
<td>8.2715 7.6346 7.70% 10.4 25s</td>
<td></td>
</tr>
<tr>
<td>155922</td>
<td>81017</td>
<td>8.2715 7.6346 7.70% 10.4 25s</td>
<td></td>
</tr>
<tr>
<td>181714</td>
<td>99222</td>
<td>8.2715 7.6346 7.70% 10.4 25s</td>
<td></td>
</tr>
<tr>
<td>209550</td>
<td>118662</td>
<td>8.2715 7.6346 7.70% 10.4 25s</td>
<td></td>
</tr>
<tr>
<td>234738</td>
<td>136907</td>
<td>8.2715 7.6346 7.70% 10.4 25s</td>
<td></td>
</tr>
<tr>
<td>262662</td>
<td>158853</td>
<td>8.2715 7.6346 7.70% 10.4 25s</td>
<td></td>
</tr>
<tr>
<td>*283256</td>
<td>33844</td>
<td>8.2715 7.6346 7.70% 10.4 25s</td>
<td></td>
</tr>
<tr>
<td>*283273</td>
<td>121603</td>
<td>8.2715 7.6346 7.70% 10.4 25s</td>
<td></td>
</tr>
<tr>
<td>*283508</td>
<td>114435</td>
<td>8.2715 7.6346 7.70% 10.4 25s</td>
<td></td>
</tr>
<tr>
<td>283524</td>
<td>114559</td>
<td>8.2715 7.6346 7.70% 10.4 25s</td>
<td></td>
</tr>
<tr>
<td>294708</td>
<td>118404</td>
<td>8.2715 7.6346 7.70% 10.4 25s</td>
<td></td>
</tr>
<tr>
<td>*317714</td>
<td>45285</td>
<td>8.2715 7.6346 7.70% 10.4 25s</td>
<td></td>
</tr>
</tbody>
</table>

Explored 317872 nodes (2919175 simplex iterations) in 73.57 seconds
Thread count was 4 (of 4 available processors)
Optimal solution found (tolerance 1.00e-04)
Best objective 7.6346078431e+00, best bound 7.6346078431e+00, gap 0.0%

---

An Extreme Case

Gurobi Optimizer version 2.0.0

Read MPS format model from file ns1671066.mps.bz2
ns167106: 316 Rows, 2840 Columns, 31418 NonZeros
Presolved: 315 Rows, 1819 Columns, 19336 Nonzeros

Root relaxation: objective 7.634608e+00, 241 iterations, 0.01 seconds

<table>
<thead>
<tr>
<th>Nodes</th>
<th>Current Node</th>
<th>Objective Bounds</th>
<th>Work</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Expl Unexpl</td>
<td>Obj Depth IntInf</td>
<td>Incumbent BestBd Gap</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>7.6346 0 20</td>
<td>7.6346 49.3589 7.6346 84.5% 0s</td>
</tr>
<tr>
<td>H 0</td>
<td>0</td>
<td>7.8698 7.6346 2.99% 0s</td>
<td></td>
</tr>
<tr>
<td>H 0</td>
<td>0</td>
<td>7.6346 7.6346 0.0% 0s</td>
<td></td>
</tr>
</tbody>
</table>

Explored 0 nodes (564 simplex iterations) in 0.12 seconds
Thread count was 4 (of 4 available processors)
Optimal solution found (tolerance 1.00e-04)
Best objective 7.6346078431e+00, best bound 7.6346078431e+00, gap 0.0%

---

Found heuristic solution: objective 152.7836
Found heuristic solution: objective 49.3589
A More Typical Example

Gurobi Optimizer version 2.0.0

Read MPS format model from file neos17.mps.bz2
NEOS17: 486 Rows, 535 Columns, 4931 NonZeros
Presolved: 486 Rows, 511 Columns, 3194 Nonzeros

Root relaxation: objective 6.814985e-04, 545 iterations, 0.01 seconds

<table>
<thead>
<tr>
<th>Nodes</th>
<th>Current Node</th>
<th>Objective Bounds</th>
<th>Work</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expl</td>
<td>Unexpl</td>
<td>Obj</td>
<td>Depth</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0.0007</td>
<td>0</td>
</tr>
<tr>
<td>H</td>
<td>0</td>
<td>0</td>
<td>0.2227</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0.0211</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0.0249</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>0.0249</td>
<td>0</td>
</tr>
<tr>
<td>H 1057</td>
<td>534</td>
<td>0.2032</td>
<td>0.0365</td>
</tr>
<tr>
<td>H 1064</td>
<td>513</td>
<td>0.1983</td>
<td>0.0374</td>
</tr>
<tr>
<td>H 1068</td>
<td>469</td>
<td>0.1836</td>
<td>0.0374</td>
</tr>
<tr>
<td>H 1784</td>
<td>396</td>
<td>0.1797</td>
<td>0.0374</td>
</tr>
<tr>
<td>H 1788</td>
<td>350</td>
<td>0.1672</td>
<td>0.0374</td>
</tr>
<tr>
<td>H 1790</td>
<td>329</td>
<td>0.1672</td>
<td>0.0374</td>
</tr>
<tr>
<td>H 1853</td>
<td>260</td>
<td>0.1503</td>
<td>0.0374</td>
</tr>
<tr>
<td>H 1928</td>
<td>225</td>
<td>0.1502</td>
<td>0.0374</td>
</tr>
<tr>
<td>H 2104 321</td>
<td>0.1500</td>
<td>0.0374</td>
<td>75.1%</td>
</tr>
<tr>
<td>8980</td>
<td>2701</td>
<td>infeasible</td>
<td>79</td>
</tr>
<tr>
<td>30632</td>
<td>5748</td>
<td>0.1493</td>
<td>159</td>
</tr>
<tr>
<td>70932</td>
<td>11195</td>
<td>infeasible</td>
<td>150</td>
</tr>
<tr>
<td>113234</td>
<td>13069</td>
<td>cutoff 93</td>
<td>0.1500</td>
</tr>
<tr>
<td>355409</td>
<td>11955</td>
<td>infeasible</td>
<td>147</td>
</tr>
<tr>
<td>197219</td>
<td>8591</td>
<td>infeasible</td>
<td>157</td>
</tr>
<tr>
<td>242763</td>
<td>4142</td>
<td>cutoff 156</td>
<td>0.1500</td>
</tr>
</tbody>
</table>

Cutting planes:
Gomory: 36

Explored 257819 nodes (2719032 simplex iterations) in 36.53 seconds
Thread count was 4 (of 4 available processors)

Optimal solution found (tolerance 1.00e-04)
Best objective 1.5000257742e-01, best bound 1.4999068902e-01, gap 0.0079%
Performance Benchmarks

• Performance test sets:
  – Mittelmann feasibility test set:
    • 34 models, difficult to find feasible solutions
    • http://plato.asu.edu/ftp/feas_bench.html

• Test platform:
  – Q9450 (2.66 GHz, quad-core system)

• Geometric Means
  – Run on a single processor
  – Gurobi 1.1 is 2.3X faster than CPLEX 12.0