MIP Models and Heuristics
MIP as a Heuristic

• Tempting to focus exclusively on optimality
  • Comforting to know that you can't find a better solution

• Typically overkill
  • Uncertainty/errors in data

• MIP often used as a heuristic
  • Lower bound is nice
  • Upper bound (feasible solution) is what you typically take away

• Trivial to use MIP solver as a heuristic
  • Just stop before proven optimal solution is found

• This session focuses on advanced techniques
  • MIP starts
  • Variable hints
  • Callbacks
Injecting Solution Information

• Three ways to inject solution information:
  • MIP Start
    • Pass a known feasible solution (or partial solution) when optimization starts
    • MIP solver will try to reproduce that solution
      • Limited repair capabilities if that solution is not feasible
  • Variable hints
    • Pass hints about promising values for variables, and relative priorities of those hints
    • Hints used in multiple phases of algorithm
      • Heuristics and branching
  • Callbacks
    • User code called at each node of branch-and-cut tree
    • Can query relaxation solution, and can inject a feasible solution (or partial solution)
Injecting Solution Information

• Three ways to inject solution information:

  • MIP Start
    • Pass a known feasible solution (or partial solution) when optimization starts
    • MIP solver will try to reproduce that solution
      • Limited repair capabilities if that solution is not feasible

  • Variable hints
    • Pass hints about promising values for variables, and relative priorities of those hints
    • Hints used in multiple phases of algorithm
      • Heuristics and branching

  • Callbacks
    • User code called at each node of branch-and-cut tree
    • Can query relaxation solution, and can inject a feasible solution (or partial solution)
Combining Solution Schemes

• Often two very different approaches to solving a problem
  • Problem-specific heuristic
  • MIP model

• Problem-specific heuristics have plusses and minuses
  • By utilizing domain information
    • Quick
    • Possibly gives higher-quality initial solution than general-purpose MIP heuristic
  • But:
    • Typically no lower bound
      • No optimality gap information
    • Difficult to implement an exhaustive search
      • No way to get a proven optimal solution
    • Often difficult to extend
      • When problem changes slightly (e.g., new type of constraint)
    • Often difficult to achieve diversity
      • Solution quality may hit a plateau quickly
MIP Start

• Simple solution:
  • Run problem-specific heuristic first
  • Feed result into MIP model as a MIP start
  • Let MIP solver continue to find
    • Lower bound
    • Better solutions
Example Application – Open-Pit Mining
Open-Pit Mining Model

- **Problem:**
  - Decide which cells to mine in each time period

- **Objective:**
  - Mine the cells with the most valuable raw materials
    - Some cells have negative value – cost more to extract than they net in raw material value

- **Constraints:**
  - Can't mine a cell until after you've mined the cells above it
    - Note: "cells", not "cell" – can't mine a vertical hole
      - Limit slope to reduce chance of a cave-in
      - Trucks need to drive down to haul out dirt
  - Limited capacity to pull dirt out of the ground per time period
    - Limited number of trucks
    - Raw material extraction facilities have limited capacity
Open-Pit Mining Model – 2-D Slice

• Visit http://examples.gurobi.com/open-pit-mining for an interactive mining example...
Open-Pit Mining Model

• Simple example 3-D mining model:
  • Variables:
    • $\text{mined}_{x,y,z,t}$: binary, determines whether cell at grid location $(x,y,z)$ has been mined at (or before) time $t$
  • Constraints:
    • Precedence:
      • Time: $\text{mined}_{x,y,z,t} \geq \text{mined}_{x,y,z,t-1}$
      • Space: $\text{mined}_{x,y,z,t} \leq \text{mined}_{x,y,z+1,t}$
        $\text{mined}_{x,y,z,t} \leq \text{mined}_{x-1,y,z+1,t}$
        $\text{mined}_{x,y,z,t} \leq \text{mined}_{x+1,y,z+1,t}$
        $\text{mined}_{x,y,z,t} \leq \text{mined}_{x,y-1,z+1,t}$
        $\text{mined}_{x,y,z,t} \leq \text{mined}_{x,y+1,z+1,t}$
    • Capacity:
      • $\sum_{x,y,z} (\text{mined}_{x,y,z,t} - \text{mined}_{x,y,z,t-1}) \leq \text{capacity}_t$
Solving the Open-Pit Mining Problem

- Default settings:

Optimize a model with 167806 rows, 33556 columns and 449282 nonzeros
Variable types: 0 continuous, 33556 integer (33556 binary)
Coefficient statistics:
  - Matrix range: [9e-01, 1e+00]
  - Objective range: [4e-07, 4e-01]
  - Bounds range: [1e+00, 1e+00]
  - RHS range: [1e+02, 1e+02]
Found heuristic solution: objective 10.7392
Presolve time: 3.34s
Presolved: 167806 rows, 33556 columns, 449282 nonzeros

Variable types: 0 continuous, 33556 integer (33556 binary)
...
Solving the Open-Pit Mining Problem

Root simplex log...

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Objective</th>
<th>Primal Inf.</th>
<th>Dual Inf.</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>3270</td>
<td>1.0033671e+03</td>
<td>0.0000000e+00</td>
<td>6.021110e+04</td>
<td>5s</td>
</tr>
<tr>
<td>15260</td>
<td>7.6003191e+02</td>
<td>0.0000000e+00</td>
<td>1.472755e+05</td>
<td>10s</td>
</tr>
<tr>
<td>26160</td>
<td>6.8283505e+02</td>
<td>0.0000000e+00</td>
<td>5.575091e+04</td>
<td>15s</td>
</tr>
<tr>
<td>36406</td>
<td>6.3195748e+02</td>
<td>0.0000000e+00</td>
<td>4.191839e+04</td>
<td>20s</td>
</tr>
<tr>
<td>44690</td>
<td>6.0376674e+02</td>
<td>0.0000000e+00</td>
<td>7.441997e+04</td>
<td>25s</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>117938</td>
<td>4.5763821e+02</td>
<td>0.0000000e+00</td>
<td>1.412679e+04</td>
<td>70s</td>
</tr>
<tr>
<td>124042</td>
<td>4.5423261e+02</td>
<td>0.0000000e+00</td>
<td>8.671474e+03</td>
<td>75s</td>
</tr>
<tr>
<td>130364</td>
<td>4.5196987e+02</td>
<td>0.0000000e+00</td>
<td>2.823627e+03</td>
<td>80s</td>
</tr>
<tr>
<td>135145</td>
<td>4.5135892e+02</td>
<td>0.0000000e+00</td>
<td>0.0000000e+00</td>
<td>84s</td>
</tr>
<tr>
<td>135145</td>
<td>4.5135892e+02</td>
<td>0.0000000e+00</td>
<td>0.0000000e+00</td>
<td>84s</td>
</tr>
</tbody>
</table>

Root relaxation: objective 4.513589e+02, 135145 iterations, 80.59 seconds
Solving the Open-Pit Mining Problem

<table>
<thead>
<tr>
<th>Expl</th>
<th>Unexpl</th>
<th>Obj</th>
<th>Depth</th>
<th>IntInf</th>
<th>Incumbent</th>
<th>BestBd</th>
<th>Gap %</th>
<th>It/Node</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>451.35892</td>
<td>0</td>
<td>996</td>
<td>10.73916</td>
<td>451.35892</td>
<td>4103%</td>
<td>-</td>
</tr>
<tr>
<td>H</td>
<td>0</td>
<td>0</td>
<td>448.0396048</td>
<td>0</td>
<td>977</td>
<td>450.38224</td>
<td>451.35892</td>
<td>0.74%</td>
<td>-</td>
</tr>
<tr>
<td>H</td>
<td>0</td>
<td>0</td>
<td>450.3822401</td>
<td>0</td>
<td>450.38224</td>
<td>451.33446</td>
<td>451.35892</td>
<td>0.22%</td>
<td>-</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>451.33446</td>
<td>0</td>
<td>977</td>
<td>450.38224</td>
<td>451.33446</td>
<td>0.21%</td>
<td>-</td>
</tr>
<tr>
<td>H</td>
<td>0</td>
<td>0</td>
<td>450.5145733</td>
<td>0</td>
<td>977</td>
<td>450.38224</td>
<td>451.33446</td>
<td>0.18%</td>
<td>-</td>
</tr>
</tbody>
</table>
MIP Start

• First MIP solution is terrible
• Exploit domain information to find a better one?
• Trivial "greedy" heuristic:
  • Repeat
    • Pick the 'exposed' cell with the largest profit (or smallest loss)
    • If we don't have sufficient capacity in this time period
      • Advance the time period t
    • Mine the cell
      • Possibly creating new 'exposed' cells

• Choose the best solution found along the way
  • Set it as a MIP start
MIP Start

• "Set it as a MIP start"
• Mechanics?

```python
# Call greedy heuristic
# Return solution in dictionary greedy_x
greedy_x = {}
greedy_heur(model, greedy_x)

# Populate 'start' attribute from greedy solution
for v in vars:
    v.start = greedy_x[v]
```
Quick Aside: Partial MIP Start

• Note: you don't need to provide start values for every variable

• Solver will perform a truncated sub-MIP solve to try to complete your start
  • Fix all variables with provided start values
  • Solve a MIP on the remaining variables
    • Using a node limit (limit controlled by `SubMIPNodes` parameter)

• Need to use some caution
  • For example, we'll accept a MIP start with only one value
  • Resulting sub-MIP can be expensive
Solving the Open-Pit Mining Problem

• With trivial heuristic:

  Presolved: 167806 rows, 33556 columns, 449282 nonzeros

  Loaded MIP start with objective 428.813

  Variable types: 0 continuous, 33556 integer (33556 binary)

  ...

• Runtime for heuristic:
  • Less than 1s
Solving the Open-Pit Mining Problem

- If you let it run for a while…

<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</td>
</tr>
<tr>
<td>...</td>
<td>H 1055   938</td>
<td>450.6810903</td>
<td>451.30920</td>
</tr>
<tr>
<td>1061</td>
<td>944</td>
<td>451.17448</td>
<td>19 779</td>
</tr>
<tr>
<td>1070</td>
<td>952</td>
<td>451.12340</td>
<td>25 642</td>
</tr>
<tr>
<td>1202</td>
<td>1069</td>
<td>451.19371</td>
<td>12 1176</td>
</tr>
<tr>
<td>1204</td>
<td>1070</td>
<td>451.27392</td>
<td>6 996</td>
</tr>
<tr>
<td>1205</td>
<td>1071</td>
<td>451.09846</td>
<td>48 1124</td>
</tr>
<tr>
<td>1206</td>
<td>1072</td>
<td>450.84933</td>
<td>78 1146</td>
</tr>
<tr>
<td>H 1206</td>
<td>1018</td>
<td>450.7981045</td>
<td>451.26019</td>
</tr>
<tr>
<td>1208</td>
<td>1019</td>
<td>451.20933</td>
<td>45 1293</td>
</tr>
<tr>
<td>1209</td>
<td>1020</td>
<td>451.09179</td>
<td>45 1265</td>
</tr>
</tbody>
</table>
Better Heuristic

• "Rolling horizon" heuristic:
  • Start from greedy heuristic solution
  • Repeat
    • Choose a contiguous set of time periods (e.g. periods 3-6)
    • Freeze mining decisions from current solution outside of this period
    • Reoptimize decisions within this period
      • As a MIP
      • May produce a better solution

• Much more expensive than greedy heuristic alone
  • Solve multiple, smaller MIPs
  • Total runtime ~60s

• Also much more effective…

  Loaded MIP start with objective 450.802
Better Heuristic

- If you let it run for a while…

<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</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>451.31249</td>
<td>0 1176 450.80167</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>451.30992</td>
<td>2 1120 450.80167</td>
</tr>
<tr>
<td>57</td>
<td>58</td>
<td>451.28197</td>
<td>16 789 450.80167</td>
</tr>
<tr>
<td>79</td>
<td>82</td>
<td>451.23186</td>
<td>21 637 450.80167</td>
</tr>
<tr>
<td>H 98</td>
<td>82</td>
<td></td>
<td></td>
</tr>
<tr>
<td>159</td>
<td>159</td>
<td>451.22172</td>
<td>41 660 450.80800</td>
</tr>
<tr>
<td>H 185</td>
<td>159</td>
<td></td>
<td></td>
</tr>
<tr>
<td>308</td>
<td>311</td>
<td>451.19004</td>
<td>69 585 450.81517</td>
</tr>
<tr>
<td>H 549</td>
<td>532</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1114</td>
<td>998</td>
<td>450.96504</td>
<td>100 1176 450.82864</td>
</tr>
<tr>
<td>1116</td>
<td>999</td>
<td>451.13381</td>
<td>34 996 450.82864</td>
</tr>
<tr>
<td>1121</td>
<td>1003</td>
<td>451.14416</td>
<td>24 1129 450.82864</td>
</tr>
</tbody>
</table>
Injecting Solution Information

• Three ways to inject solution information:
  
  • MIP Start
    • Pass a known feasible solution (or partial solution) when optimization starts
    • MIP solver will try to reproduce that solution
      • Limited repair capabilities if that solution is not feasible
  
  • Variable hints
    • Pass hints about promising values for variables, and relative priorities of those hints
    • Hints used in multiple phases of algorithm
      • Heuristics and branching
  
  • Callbacks
    • User code called at each node of branch-and-cut tree
    • Can query relaxation solution, and can inject a feasible solution (or partial solution)
Variable Hints – Use Cases

• Sliding time window
  • Model solves for a window of time \( t=0,1,2,\ldots,n \)
  • Given a solution for \( t=0,1,2,\ldots,n \):
    • Deploy solution for \( t=0 \)
    • Gather new measured data
    • Create updated model for \( t=1,2,\ldots,n+1 \)
  • Can use \( t=1,2,\ldots,n \) solution from first model as hint for next model
Variable Hints – Use Cases

• Multiple scenarios
  • Solve multiple variants of the same model
  • Small perturbation to obj, RHS, etc.
  • Often lots of overlap between high-quality solutions
    • Small perturbation won't completely change the character of the solution
  • Use solutions from other scenarios as hints
Variable Hints – Multiple Scenario Example

• Read a difficult model from a file
• Solve it 10 times with perturbed objectives  
  • Count # times each binary variable takes value 0/1
• Use more common value as hint value  
  • # of times it takes that value as hint priority
m = read('ljb12')
perturb = 1.2
for i in range(REPS):
    # perturb objective
    for v in binaries:
        v.obj = random.uniform(1/perturb,perturb)*v.obj +
        random.uniform(-1e-4,1e-4)

m.reset()
m.optimize()

# adjust counts
for v in binaries:
    val = int(round(v.x))
    count[v][val] = count[v][val]+1
for i in range(REPS):
    # perturb objective
...
# solve without hints
...
# solve with hints
m.reset()
for v in binaries:
    if count[v][0] > count[v][1]:
        v.varhintval = 0
        v.varhintpri = count[v][0]
    elif count[v][0] < count[v][1]:
        v.varhintval = 1
        v.varhintpri = count[v][1]
    m.optimize()
Variable Hints – Multiple Scenario Example

• Using 10 second time limit for 'training' runs and 1 second time limit for tests

<table>
<thead>
<tr>
<th>Trial</th>
<th>no hint</th>
<th>obj: 1.00000e+100</th>
<th>hint obj: 5.90331e+00</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trial 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Trial 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Trial 2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Trial 3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Trial 4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Trial 5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Trial 6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Trial 7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Trial 8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Trial 9</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Variable Hints – Multiple Scenario Example

• Using 20 second time limit for 'training' runs and 2 second time limit for tests

<table>
<thead>
<tr>
<th>Trial</th>
<th>no hint</th>
<th>hint</th>
<th>Trial</th>
<th>no hint</th>
<th>hint</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>obj: 6.38623e+00</td>
<td>hint obj: 5.88125e+00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>obj: 6.38623e+00</td>
<td>hint obj: 5.80083e+00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>obj: 6.38623e+00</td>
<td>hint obj: 5.76273e+00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>obj: 6.38623e+00</td>
<td>hint obj: 5.78522e+00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>obj: 6.38623e+00</td>
<td>hint obj: 5.87428e+00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>obj: 6.38623e+00</td>
<td>hint obj: 5.79409e+00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>obj: 6.38623e+00</td>
<td>hint obj: 5.74034e+00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>obj: 6.15425e+00</td>
<td>hint obj: 5.84347e+00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>obj: 6.38623e+00</td>
<td>hint obj: 5.76325e+00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>obj: 6.38623e+00</td>
<td>hint obj: 5.73973e+00</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Variable Hints – Multiple Scenario Example

- What are hints doing...?

Root relaxation: objective \(-5.31377\times10^{-1}\), 2533 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>-0.53114</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>-0.23949</td>
<td>0</td>
</tr>
</tbody>
</table>

New incumbent: VarHint heuristic

H 0 0 5.8114413 -0.23949 104% - 1s
0 0 -0.22284 0 2145 5.81144 -0.22284 104% - 1s
Variable Hints – Multiple Scenario Example

• Note: this isn't actually that effective of a strategy in general
  • But it is extremely effective on some models

• Key point
  • If you know something about what good solutions look like, try using variable hints to pass this info to us
Injecting Solution Information

• Three ways to inject solution information:
  • MIP Start
    • Pass a known feasible solution (or partial solution) when optimization starts
    • MIP solver will try to reproduce that solution
      • Limited repair capabilities if that solution is not feasible
  • Variable hints
    • Pass hints about promising values for variables, and relative priorities of those hints
    • Hints used in multiple phases of algorithm
      • Heuristics and branching
  • Callbacks
    • User code called at each node of branch-and-cut tree
    • Can query relaxation solution, and can inject a feasible solution (or partial solution)
Solution Callback

• At each node in B&B search…
  • User routine is called, and can query…
    • Node relaxation solution
    • New feasible solution
  • Can return a solution (or partial solution)
Open-Pit Mining Revisited

• Return to open-pit mining example

• Original greedy heuristic:
  • Choose exposed cells based on objective value
  • Doesn't require a relaxation solution

• New greedy heuristic:
  • Choose exposed cells based on relaxation value
  • Uses LP solution to choose promising cells
    • Much less "greedy" – LP looks ahead in time

• Can run it at every node, every 10\textsuperscript{th} node, etc.
Open-Pit Mining Revisited

• Results:
  • Tried many different variants
  • Quite good at finding ‘good’ solutions
  • Doesn’t find better solutions

• General MIP heuristics are quite effective
  • Don't expect to be able to beat them very often
Thank you – Questions?