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)

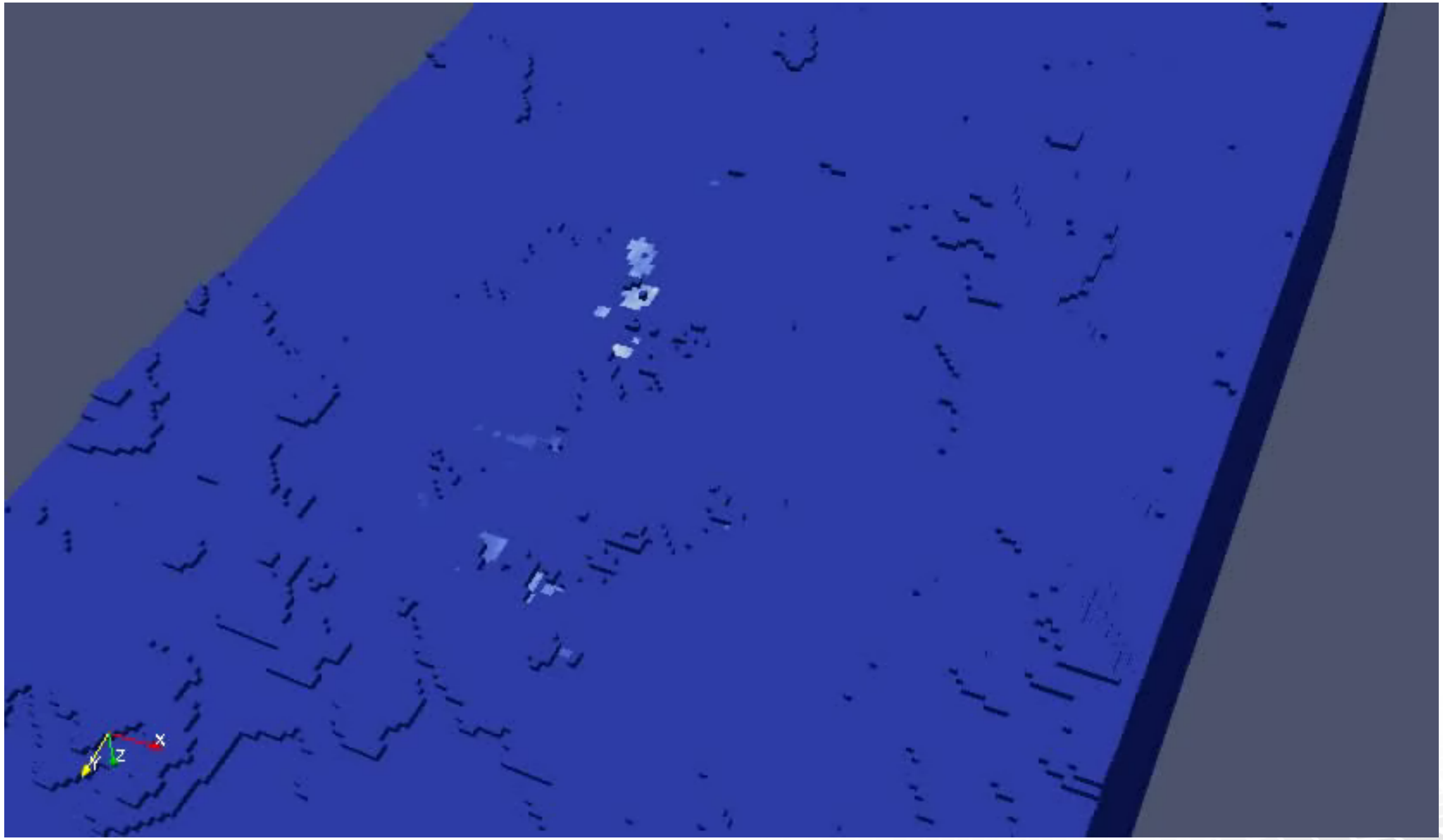
Combining Solution Schemes

- 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



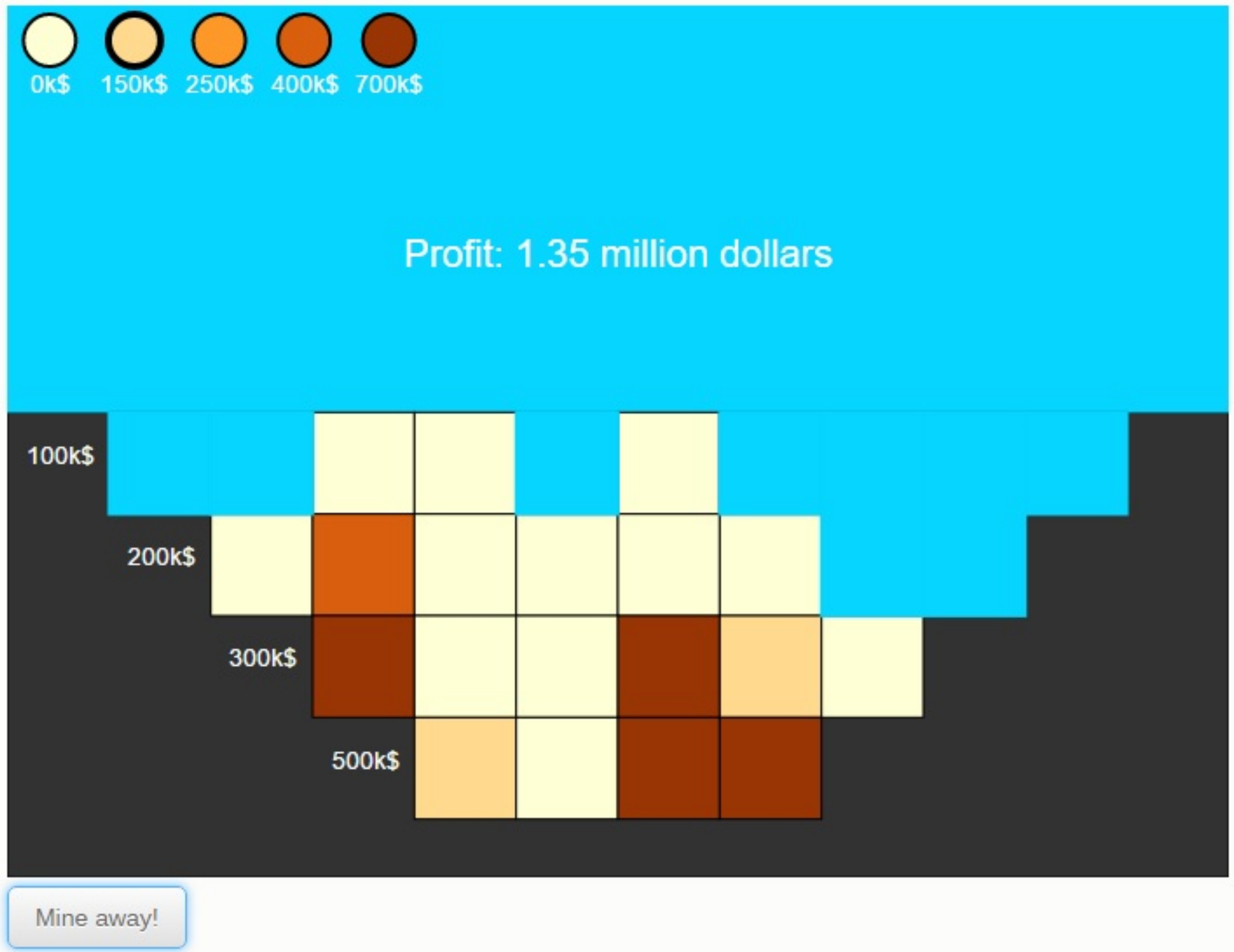
Open-Pit Mining Model

GUROBI OPTIMIZATION

- 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

GUROBI OPTIMIZATION

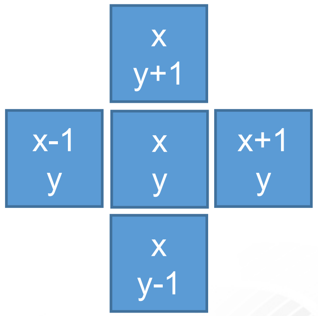
- Simple example 3-D mining model:
  - Variables:
    - $mined_{x,y,z,t}$: binary, determines whether cell at grid location (x,y,z) has been mined at (or before) time t
  - Constraints:
    - Precedence:
      - Time: $mined_{x,y,z,t} \geq mined_{x,y,z,t-1}$
      - Space: $mined_{x,y,z,t} \leq mined_{x,y,z+1,t}$

        $mined_{x,y,z,t} \leq mined_{x-1,y,z+1,t}$

        $mined_{x,y,z,t} \leq mined_{x+1,y,z+1,t}$

        $mined_{x,y,z,t} \leq mined_{x,y-1,z+1,t}$

        $mined_{x,y,z,t} \leq mined_{x,y+1,z+1,t}$



    - Capacity:
      - $sum_{x,y,z}\ (mined_{x,y,z,t} - mined_{x,y,z,t-1}) \leq capacity_t$

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...

Iteration    Objective       Primal Inf.    Dual Inf.      Time
     3270    1.0033671e+03   0.000000e+00   6.021110e+04      5s
    15260    7.6003191e+02   0.000000e+00   1.472755e+05     10s
    26160    6.8283505e+02   0.000000e+00   5.575091e+04     15s
    36406    6.3195748e+02   0.000000e+00   4.191839e+04     20s
    44690    6.0376674e+02   0.000000e+00   7.441997e+04     25s
...
   117938    4.5763821e+02   0.000000e+00   1.412679e+04     70s
   124042    4.5423261e+02   0.000000e+00   8.671474e+03     75s
   130364    4.5196987e+02   0.000000e+00   2.823627e+03     80s
   135145    4.5135892e+02   0.000000e+00   0.000000e+00     84s
   135145    4.5135892e+02   0.000000e+00   0.000000e+00     84s

Root relaxation: objective 4.513589e+02, 135145 iterations, 80.59 seconds
```

Solving the Open-Pit Mining Problem

GUROBI OPTIMIZATION

```
     Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

      0     0  451.35892    0  996   10.73916  451.35892  4103%     -   85s
H     0     0                      448.0396048  451.35892  0.74%     -   93s
H     0     0                      450.3822401  451.35892  0.22%     -   99s
      0     0  451.33446    0  977  450.38224  451.33446  0.21%     -  106s
H     0     0                      450.5145733  451.33446  0.18%     -  112s
```

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?

```
# 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…

```
    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time
...
H 1055   938                       450.6810903  451.30920  0.14%  33.7  303s
  1061   944  451.17448   19  779   450.68109  451.30920  0.14%  34.8  307s
  1070   952  451.12340   25  642   450.68109  451.30920  0.14%  35.1  311s
  1202  1069  451.19371   12 1176   450.68109  451.30920  0.14%  35.3  374s
  1204  1070  451.27392    6  996   450.68109  451.27392  0.13%  35.3  419s
  1205  1071  451.09846   48 1124   450.68109  451.26803  0.13%  35.2  446s
  1206  1072  450.84933   78 1146   450.68109  451.26775  0.13%  35.2  457s
H 1206  1018                       450.7981045  451.26019  0.10%  35.2  482s
  1208  1019  451.20933   45 1293   450.79810  451.25898  0.10%  35.1  489s
  1209  1020  451.09179   45 1265   450.79810  451.25705  0.10%  35.1  500s
```

Better Heuristic

GUROBI OPTIMIZATION

- "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…

```
    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time
...
     0     2  451.31249    0 1176  450.80167  451.31249  0.11%     -  219s
     3     8  451.30992    2 1120  450.80167  451.31154  0.11%  12.7  220s
    57    58  451.28197   16  789  450.80167  451.30960  0.11%  26.8  227s
    79    82  451.23186   21  637  450.80167  451.30960  0.11%  51.6  232s
H   98    82                    450.8080003  451.30960  0.11%  43.5  233s
   159   159  451.22172   41  660  450.80800  451.30960  0.11%  31.9  238s
H  185   159                    450.8151686  451.30960  0.11%  30.3  238s
   308   311  451.19004   69  585  450.81517  451.30960  0.11%  27.1  244s
H  549   532                    450.8286395  451.30960  0.11%  24.3  253s
  1114   998  450.96504  100 1176  450.82864  451.30762  0.11%  24.8  342s
  1116   999  451.13381   34  996  450.82864  451.27350  0.10%  24.7  370s
  1121  1003  451.14416   24 1129  450.82864  451.24033  0.09%  24.6  421s
```

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,…,n)
  - Given a solution for t=0…n:
    - Deploy solution for t=0
    - Gather new measured data
    - Create updated model for t=1…n+1
  - Can use t=1…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

Variable Hints – Multiple Scenario Example

```
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
```

Variable Hints – Multiple Scenario Example

```
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

```
Trial 0   no hint obj: 1.00000e+100   hint obj: 5.90331e+00
Trial 1   no hint obj: 1.00000e+100   hint obj: 5.95868e+00
Trial 2   no hint obj: 1.00000e+100   hint obj: 5.93106e+00
Trial 3   no hint obj: 1.00000e+100   hint obj: 6.31872e+00
Trial 4   no hint obj: 1.00000e+100   hint obj: 6.02026e+00
Trial 5   no hint obj: 1.00000e+100   hint obj: 5.95413e+00
Trial 6   no hint obj: 1.00000e+100   hint obj: 5.93878e+00
Trial 7   no hint obj: 1.00000e+100   hint obj: 5.97493e+00
Trial 8   no hint obj: 1.00000e+100   hint obj: 6.38623e+00
Trial 9   no hint obj: 1.00000e+100   hint obj: 6.11830e+00
```

Variable Hints – Multiple Scenario Example

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

```
Trial 0   no hint obj: 6.38623e+00   hint obj: 5.88125e+00
Trial 1   no hint obj: 6.38623e+00   hint obj: 5.80083e+00
Trial 2   no hint obj: 6.38623e+00   hint obj: 5.76273e+00
Trial 3   no hint obj: 6.38623e+00   hint obj: 5.78522e+00
Trial 4   no hint obj: 6.38623e+00   hint obj: 5.87428e+00
Trial 5   no hint obj: 6.38623e+00   hint obj: 5.79409e+00
Trial 6   no hint obj: 6.38623e+00   hint obj: 5.74034e+00
Trial 7   no hint obj: 6.15425e+00   hint obj: 5.84347e+00
Trial 8   no hint obj: 6.38623e+00   hint obj: 5.76325e+00
Trial 9   no hint obj: 6.38623e+00   hint obj: 5.73973e+00
```

Variable Hints – Multiple Scenario Example

- What are hints doing…?

```
Root relaxation: objective -5.311377e-01, 2533 iterations, 0.01 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0   -0.53114    0 2424          -   -0.53114      -     -    0s
     0     0   -0.23949    0 2145          -   -0.23949   104%     -    0s
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^{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?