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:
  - $a_k x_k + \inf (\sum_{j!=k} a_j x_j) \leq \sum a_j x_j \leq b$
  - $x_k \leq (b - \inf (\sum_{j!=k} a_j x_j)) / a_k$
    - 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 $|x - x^*| <= k$
    b. Solve this MIP
    c. Replace the added constraint by $|x – x^*| >= k + 1$
    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*| <= 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

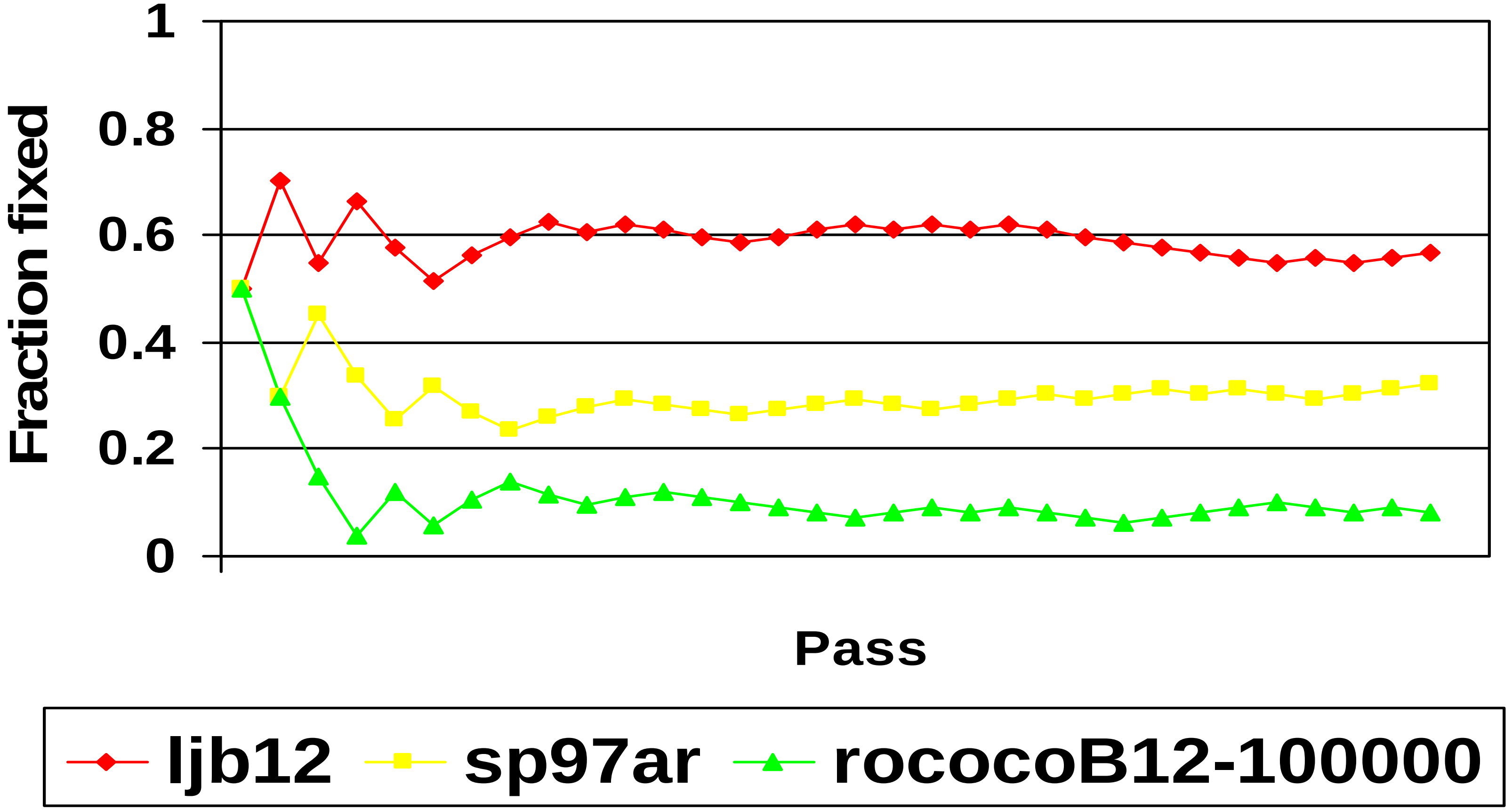
RINS

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

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

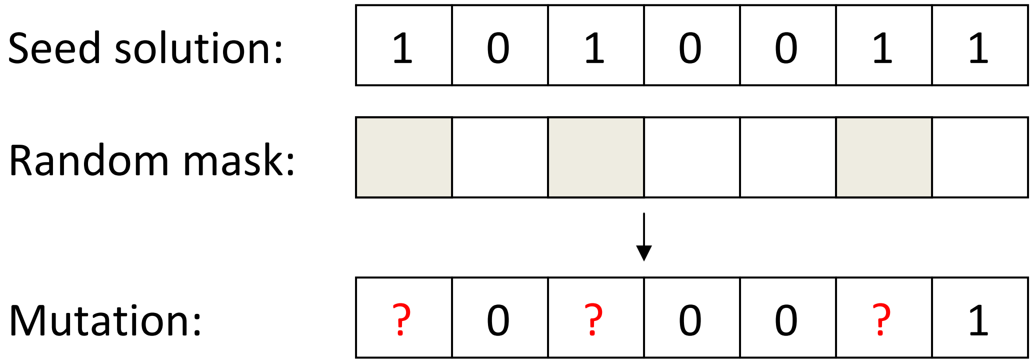
The Population

- 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:



- **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**

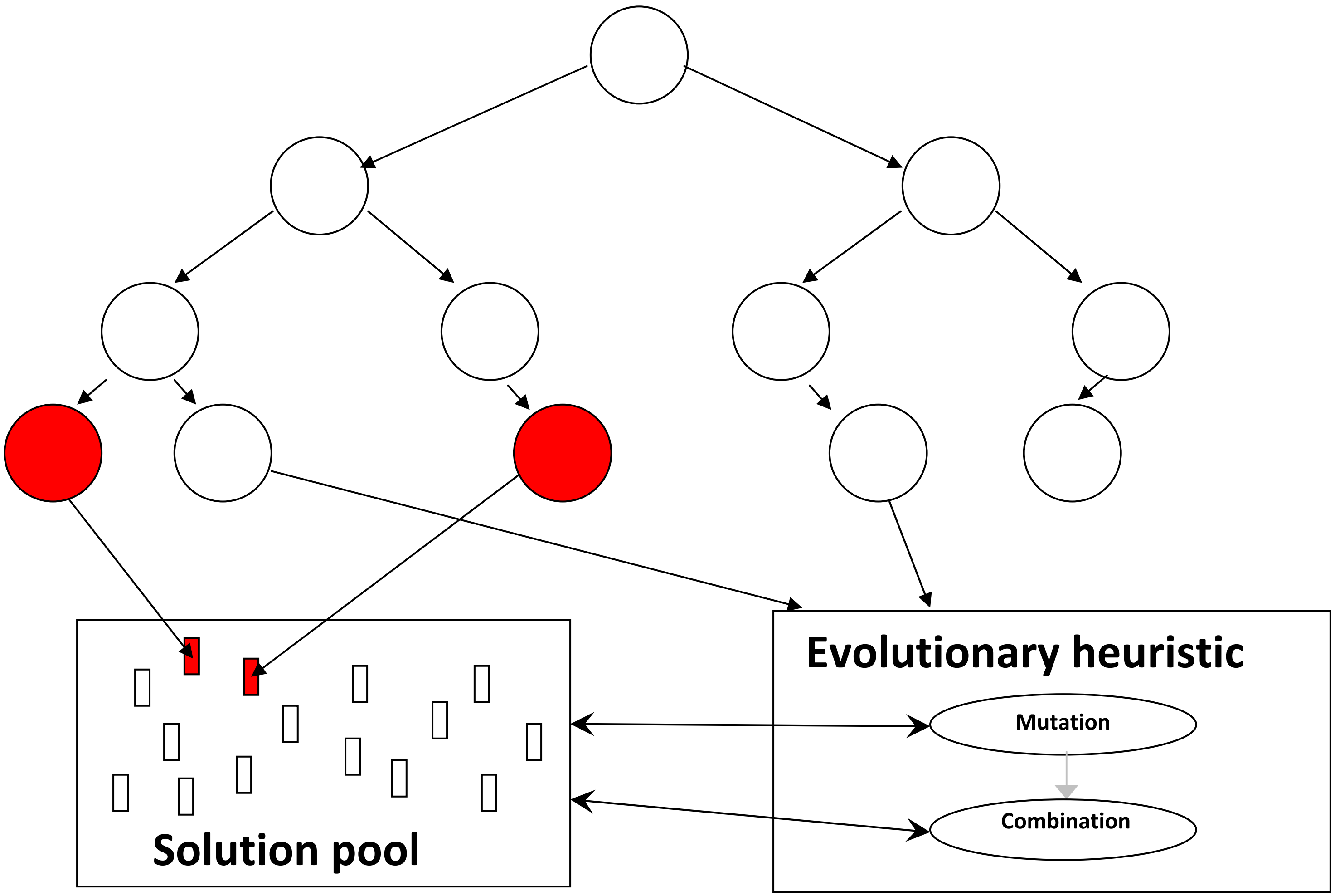
Solution Polishing

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



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

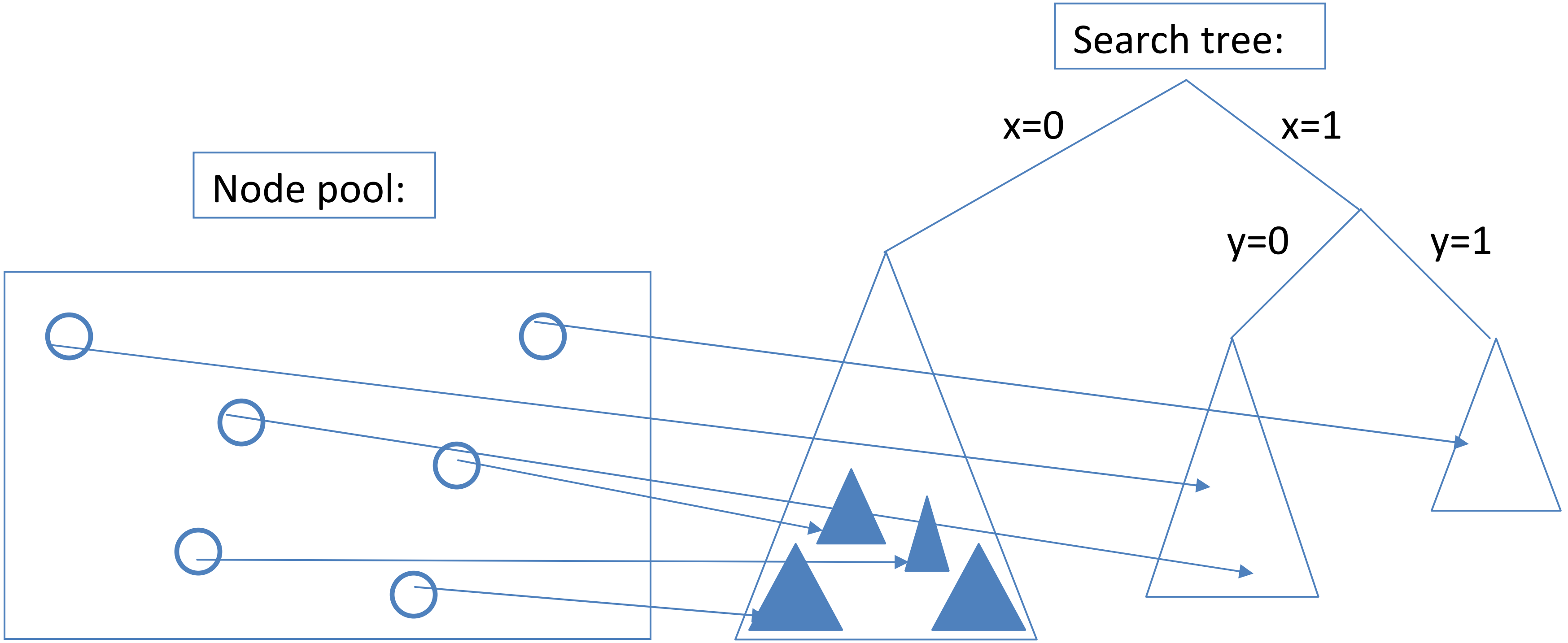
Why is it so Effective?

- 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

An Extreme Case

```
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

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

     0       0    7.6346    0   20          -   7.6346     -       -    0s
     0       0    7.6346    0   34          -   7.6346     -       -    0s
     0       0    7.6346    0    2          -   7.6346     -       -    0s
     0       0    7.6346    0   25          -   7.6346     -       -    0s
     0       0    7.6346    0    6          -   7.6346     -       -    0s
     0       2    7.6346    0    6          -   7.6346     -       -    0s
* 1998    1716            326         9.1334   7.6346  16.4%  25.2    1s
* 2002    1710            328         9.1031   7.6346  16.1%  25.1    1s
* 2172    1359            397         8.3611   7.6346  8.69%  23.9    1s
* 2177    1358            399         8.3608   7.6346  8.69%  23.8    1s
  4467    2736    7.6346  166   25    8.3608   7.6346  8.69%  23.0    5s
* 5695    3015            352         8.3453   7.6346  8.52%  20.7    5s
 23241   15991    8.3380  293   33    8.3453   7.6346  8.52%  13.7   10s
 47601   35137    7.6346   68   35    8.3453   7.6346  8.52%  11.1   15s
*55945   37046            413         8.2735   7.6346  7.72%  10.6   16s
*70873   48462            408         8.2724   7.6346  7.71%  10.1   19s
*71445   48891            442         8.2715   7.6346  7.70%  10.1   19s
 72961   50242    7.9725  114   40    8.2715   7.6346  7.70%  10.0   20s
 91853   64329    8.0481  114   24    8.2715   7.6346  7.70%  10.4   25s
*97820   47515            348         8.0819   7.6346  5.53%  10.5   26s
 111094  57352    7.6701  243   36    8.0819   7.6346  5.53%  10.6   30s
*125331  58815            336         8.0323   7.6346  4.95%  10.6   33s
 133884  65918    7.7448  191   34    8.0323   7.6346  4.95%  10.3   35s
 155922  81017    7.9642  164   57    8.0323   7.6346  4.95%  10.3   40s
 181714  99222    cutoff  210         8.0323   7.6346  4.95%  10.1   45s
 209550 118662    7.7712  201   54    8.0323   7.6346  4.95%   9.9   50s
 234738 136907    7.6723  122   55    8.0323   7.6346  4.95%   9.7   55s
 262662 156853    cutoff  170         8.0323   7.6346  4.95%   9.5   60s
*283256 133044            297         7.9649   7.6346  4.15%   9.3   63s
*283273 121603            306         7.9372   7.6346  3.81%   9.3   63s
*283308 114435            313         7.9198   7.6346  3.60%   9.3   63s
 283524 114559    7.6346   42   41    7.9198   7.6346  3.60%   9.3   65s
 294708 118404    7.8606  175   28    7.9198   7.6346  3.60%   9.4   70s
*317714  45285            267         7.7546   7.6346  1.55%   9.2   73s

Explored 317872 nodes (2918681 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%
```

```
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

Found heuristic solution: objective 152.7836
Found heuristic solution: objective 49.3589

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

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

     0     0     7.6346    0   20    49.3589     7.6346  84.5%     -    0s
H    0     0                          7.8698     7.6346  2.99%     -    0s
H    0     0                          7.6346     7.6346   0.0%     -    0s

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%
```

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

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

     0     0     0.0007    0  171          -     0.0007     -      -    0s
H    0     0                          0.2227     0.0007   100%     -    0s
     0     0     0.0211    0  171     0.2227     0.0211  90.5%     -    0s
     0     0     0.0249    0  203     0.2227     0.0249  88.8%     -    0s
     0     2     0.0249    0  203     0.2227     0.0249  88.8%     -    0s
H 1057   534                          0.2032     0.0365  82.1%  39.6    1s
H 1064   513                          0.1983     0.0374  81.2%  39.9    1s
H 1068   469                          0.1836     0.0374  79.6%  39.9    1s
H 1784   396                          0.1797     0.0374  79.2%  37.3    1s
H 1788   350                          0.1672     0.0374  77.6%  37.2    1s
H 1790   329                          0.1672     0.0374  77.6%  37.2    1s
H 1853   260                          0.1503     0.0374  75.1%  36.9    1s
H 1928   225                          0.1502     0.0374  75.1%  36.3    1s
H 2104   321                          0.1500     0.0374  75.1%  33.9    2s
  8980  2701 infeasible   79          0.1500     0.1207  19.5%  25.0    5s
 30632  5748     0.1493  159   12     0.1500     0.1428  4.77%  20.3   10s
 70932 11195 infeasible  150          0.1500     0.1454  3.05%  14.6   15s
113234 13069     cutoff   93          0.1500     0.1467  2.21%  12.8   20s
155409 11595 infeasible  147          0.1500     0.1475  1.64%  11.9   25s
197219  8591 infeasible  157          0.1500     0.1482  1.21%  11.4   30s
242763  4142     cutoff  156          0.1500     0.1491  0.63%  10.8   35s

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