Notes on MIPLIB and Benchmarking

Thorsten Koch
Why are we benchmarking?

► If you have a particular problem to solve, it typically belongs to exactly one class of instances!
► Benchmarking is mostly useful for those who develop solvers.
► It is a projection of a multidimensional vector into a 1-dimensional one. Something will be lost.
**What is the MIPLIB**

<table>
<thead>
<tr>
<th>Version</th>
<th>Year</th>
<th>#</th>
<th>Who</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1991</td>
<td>61</td>
<td>Bixby, Boyd, Indovina</td>
<td></td>
</tr>
</tbody>
</table>

The MIPLIB is a diverse collection of challenging real-world mixed integer programming (MIP) instances from various academic and industrial applications suited for benchmarking and testing of MIP solution algorithms.

The *Benchmark* set consist only of instances that could be solved in 2010 within two hours on a high-end personal computer by at least two MIP solvers.

MIPLIB 5 still contains 2 instances from Version 2, 7 instances from Version 3, and 17 instances from Version 4.
Development of MIPLIB 2003

- **easy** can be solved within an hour on a contemporary pc with a state-of-the-art solver
- **hard** are solvable but take a longer time or require specialized algorithms
- **open, i.e. not solved** instances for which the optimal solution is not known
What is important in benchmarking?

- Testset (size and selection)
- Time limit
- How to deal with instances that hit the time limit? (One/some/all solvers?)
- What is the competition exactly? (mipgap=0)
- How to deal with wrong results?
- How to combine results from instances to overall result?
- Memory limit
- Computational environment (CPU, Cores, NUMA, etc.)
LocalSolver wins on some of the hardest MIPLIB instances...

Below are some results of LocalSolver on some of the hardest MIPLIB instances. As you will notice by looking to the results (marked by *), LocalSolver wins against both Gurobi and IBM ILOG Cplex, the fastest commercial mixed-integer linear programming (MIP) solvers.

- Lower objective is better
- 5 minutes of running time for each solver
- Standard computer: Intel Core i7-820QM (4 cores, 1.73 GHz, 6 GB RAM, 8 MB cache)
- Default settings for each solver
- MIP-oriented models: not suitable for LocalSolver

Select the 21 out of 361 instances you like

From the classes you like take a few more

Chose a convenient time limit.

For many instances, the conclusion is that none of the fastest MIP solvers is currently able to provide high-quality solutions quickly in short running times, as it is needed today in the practice of optimization and operations research.

More details on instances can be found on MIPLIB 2010.
LocalSolver wins on some of the hardest MIPLIB instances...

Below are some results of LocalSolver on some of the hardest MIPLIB instances. As you will notice by looking at the results (marked by *), LocalSolver wins against both Gurobi and IBM ILOG Cplex, the fastest commercial mixed-integer linear programming (MIP) solvers:

- Lower objective is better
- 5 minutes of running time for each solver
- Standard computer: Intel Core i7-820QM (4 cores, 1.73 GHz, 6 GB RAM, 8 MB cache)
- Default settings for each solver
- MIP-oriented models: not suitable for LocalSolver

<table>
<thead>
<tr>
<th>Instances</th>
<th>Status</th>
<th>Variables</th>
<th>LocalSolver 3.1</th>
<th>Gurobi 5.5</th>
<th>Cplex 12.4</th>
<th>Optimum</th>
</tr>
</thead>
<tbody>
<tr>
<td>opm2-z10-s2</td>
<td>hard</td>
<td>6,250</td>
<td>* -25,719</td>
<td>-19,601</td>
<td>-18,539</td>
<td>-33,826</td>
</tr>
<tr>
<td>opm2-z11-s8</td>
<td>hard</td>
<td>8,019</td>
<td>* -33,028</td>
<td>-21,661</td>
<td>-18,883</td>
<td>-43,495</td>
</tr>
<tr>
<td>opm2-z12-s14</td>
<td>hard</td>
<td>10,800</td>
<td>* -46,957</td>
<td>-11,994</td>
<td>-35,469</td>
<td>-64,291</td>
</tr>
<tr>
<td>opm2-z12-s7</td>
<td>hard</td>
<td>10,800</td>
<td>* -46,034</td>
<td>-12,375</td>
<td>-30,887</td>
<td>-65,514</td>
</tr>
<tr>
<td>dc11</td>
<td>open</td>
<td>37,297</td>
<td>11,100,000</td>
<td>21,300,000</td>
<td>1,840,402</td>
<td>unknown</td>
</tr>
<tr>
<td>ds-big</td>
<td>open</td>
<td>6,020</td>
<td>9,844</td>
<td>62,520</td>
<td>5,256</td>
<td>unknown</td>
</tr>
<tr>
<td>ex1010-pi</td>
<td>open</td>
<td>25,200</td>
<td>249</td>
<td>251</td>
<td>247</td>
<td>unknown</td>
</tr>
<tr>
<td>ivu06-big</td>
<td>open</td>
<td>1,812,044</td>
<td>* 479</td>
<td>9,416</td>
<td>678</td>
<td>unknown</td>
</tr>
<tr>
<td>ivu52</td>
<td>open</td>
<td>1,423,438</td>
<td>4,907</td>
<td>16,880</td>
<td>3,285</td>
<td>unknown</td>
</tr>
<tr>
<td>mining</td>
<td>open</td>
<td>753,404</td>
<td>* -65,720,600</td>
<td>902,969,000</td>
<td>no solution</td>
<td>unknown</td>
</tr>
<tr>
<td>ns-1853823</td>
<td>open</td>
<td>213,440</td>
<td>* 2,820,000</td>
<td>4,670,000</td>
<td>no solution</td>
<td>unknown</td>
</tr>
<tr>
<td>pb-simp-nonunif</td>
<td>open</td>
<td>23,848</td>
<td>* 90</td>
<td>140</td>
<td>94</td>
<td>unknown</td>
</tr>
<tr>
<td>ramos3</td>
<td>open</td>
<td>2,187</td>
<td>* 223</td>
<td>274</td>
<td>267</td>
<td>unknown</td>
</tr>
<tr>
<td>rmr1e14</td>
<td>open</td>
<td>32,205</td>
<td>* -3469</td>
<td>-170</td>
<td>-968</td>
<td>unknown</td>
</tr>
<tr>
<td>rmr21</td>
<td>open</td>
<td>162,547</td>
<td>* -3657</td>
<td>-184</td>
<td>no solution</td>
<td>unknown</td>
</tr>
<tr>
<td>rmr25</td>
<td>open</td>
<td>326,599</td>
<td>* -3052</td>
<td>-161</td>
<td>no solution</td>
<td>unknown</td>
</tr>
<tr>
<td>sienal</td>
<td>open</td>
<td>13,741</td>
<td>256,620,000</td>
<td>315,186,152</td>
<td>54,820,419</td>
<td>unknown</td>
</tr>
<tr>
<td>st505</td>
<td>open</td>
<td>405</td>
<td>342</td>
<td>342</td>
<td>354</td>
<td>unknown</td>
</tr>
<tr>
<td>st5729</td>
<td>open</td>
<td>709</td>
<td>648</td>
<td>648</td>
<td>665</td>
<td>unknown</td>
</tr>
</tbody>
</table>

For many instances, the conclusion is that none of the fastest MIP solver is currently able to provide high-quality solutions quickly in short running times, as it is needed today in the practice of optimization and operations research.

More details on instances can be found on MIPLIB 2010.
Performance variation due to # of threads

Fig. 4: Example of performance variability depending on the number of threads. Instance roll3000 on a 32 core computer. Filled bar indicates minimum
Performance variation due to permutation

Fig. 3: Solution times for 100 permutations

(a) Instance ex9
(b) Instance pg5_34
(c) Instance neos13
(d) Instance bnatt350
(e) Instance enlight13
B&B nodes

Total number of B&B nodes processed by Gurobi 4.5.0

Thorsten Koch
Notes on MIPLINB and how to cheat on benchmarks
How to compare MIP solver performance?

There are two components to it:

- Ability to solve an instance (within some timeframe)
- Time to solve

You have always to see both numbers!

If two solvers are compared there are two possibilities:

1. Both can solve all instances.
2. There are instances only one solver can solve
3. There are instances both cannot solve
Geometric mean of results taken from the homepage of Hans Mittelmann (11. Aug 2012)
http://plato.asu.edu/ftp/milpc.html

Unsolved or failed instances are accounted for with the time limit of 1 hour.
1998 Benchmark Set

Seconds

- SIP 1.1 [CPLEX 5.0 for LP] UltraSparc 167 MHz
- CPLEX 5.0 UltraSparc 167 MHz
- SIP 1.1 [CPLEX 5.0.1 for LP] X5260 3333 MHz
- SCIP 2.1.1 [CPLEX 12.4 for LP] X5260 3333 MHz
- CPLEX 12.4 X5260 3333 MHz

Instances sorted by solution time
2012 Performance on MIPLIB 2010 / 1 thread

Instances sorted by solution time

---

CBC/1
CPLEX/1
Gurobi/1
SCIP/spx/1
XPRESS/1
Best/1
### Progress in Numbers

<table>
<thead>
<tr>
<th></th>
<th>2011</th>
<th>2012</th>
<th>Speed-up</th>
</tr>
</thead>
<tbody>
<tr>
<td>Best/1 geom. mean time [s]</td>
<td>105</td>
<td>81</td>
<td>23%</td>
</tr>
<tr>
<td>Best/12 geom. mean time [s]</td>
<td>34</td>
<td>28</td>
<td>17%</td>
</tr>
<tr>
<td>Best/1 not optimal [#]</td>
<td>4</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Best/12 not optimal [#]</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Speedup 1 to 12 [×]</td>
<td>3.1</td>
<td>2.8</td>
<td></td>
</tr>
<tr>
<td>Max(C,G,X)/Min(C,G,X) 1 thr.</td>
<td>600</td>
<td>259</td>
<td></td>
</tr>
<tr>
<td>Max(C,G,X)/Min(C,G,X) 12 thr.</td>
<td>1138</td>
<td>107</td>
<td></td>
</tr>
<tr>
<td>Best/1 geom. mean nodes [#]</td>
<td>2108</td>
<td>1958</td>
<td>8%</td>
</tr>
<tr>
<td>Best/12 geom. mean nodes [#]</td>
<td>2570</td>
<td>2406</td>
<td>7%</td>
</tr>
<tr>
<td>Max(C,G,X)/Min(C,G,X) 1 thr.</td>
<td>40,804</td>
<td>32,600</td>
<td></td>
</tr>
<tr>
<td>Max(C,G,X)/Min(C,G,X) 12 thr.</td>
<td>10,499</td>
<td>887,849</td>
<td></td>
</tr>
</tbody>
</table>

Single solvers have speed-ups up to 42%
Thorsten Koch
Notes on MIPLINB and how to cheat on benchmarks
Speedup Best/12 2011:2012 per instance

Thorsten Koch
Notes on MIPLINB and how to cheat on benchmarks
What is important in benchmarking?

- Testset (size and selection) -> Large and diverse, permuted
- Time limit -> Find quiescence point
- How to deal with instances that hit the time limit? (One/some/all solvers?) -> Include all
- How to deal with wrong results? Mark them!
- How to combine results from instances to overall result? Geometric mean!
- Memory limit -> Should not make a big difference between the solvers.
- Computational environment (CPU, Cores, NUMA, etc.) Carefully check what was selected.
Thank you very much!