Recent improvements in SAS/OR Optimization Solvers

Imre Pólik
Philipp M. Christophel, Menal Güzelsoy et al.

SAS Institute
Cary, NC, USA

INFORMS 2013
Outline

1 Symmetry

2 Parallel Tree Search

3 Other Improvements

4 Results
Symmetries in MILP

Matrix Symmetry

A symmetry of a matrix is a permutation of rows and columns such that the permuted matrix is identical to the original one.

Example

\[ \pi_1 : (c_1, c_2)(r_1, r_3) \]

\[
\begin{array}{ccc}
0 & 1 & 1 \\
1 & 1 & 0 \\
1 & 0 & 1
\end{array} \quad \begin{array}{ccc}
1 & 0 & 1 \\
1 & 1 & 0 \\
0 & 1 & 1
\end{array} \quad \begin{array}{ccc}
0 & 1 & 1 \\
1 & 1 & 0 \\
1 & 0 & 1
\end{array}
\]
Symmetries in MILP

Matrix Symmetry
A symmetry of a matrix is a permutation of rows and columns such that the permuted matrix is identical to the original one.

Constraint Symmetry
- Partition the rows by right hand side and constraint type
- Partition the columns by bounds and variable type
- Applying it to an LP/MILP solution maintains feasibility

Formulation Symmetry
- Partition the rows by right hand side and constraint type
- Partition the columns by bounds, variable type and objective coefficients
- Applying it to an LP/MILP solution maintains feasibility and objective value
### Orbit

Two variables $x_1$ and $x_2$ are in the same orbit $\theta$ if there exists a symmetry that maps $x_1$ to $x_2$.

### Orbit partition

The column/row orbit partition $O$ partitions the columns/rows by their orbits

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

- $\theta_1$
- $\theta_2$
- $\theta_3$
Symmetries in MILP

**Orbit**

Two variables $x_1$ and $x_2$ are in the same orbit $\theta$ if there exists a symmetry that maps $x_1$ to $x_2$.

**Orbit partition**

The column/row orbit partition $\mathcal{O}$ partitions the columns/rows by their orbits.

**Refinement**

Columns/rows in the same orbit have the same
- set of nonzeros (number of nonzeros)
- number of neighbours from a given orbit
Symmetry Detection – Details

Implementation
- Well-understood for graphs, matrices have special requirements
- Based on papers about SAUCY
- Handles problems with general variables and coefficients
- Node-coloured, edge-weighted, bipartite graphs
- No storage overhead, works on the matrix
- Targeted orbit computation (early stopping)
- Can be warmstarted

Usage
- Orbital branching
- Orbital fixing
- Heuristics
Symmetry Detection – Fixings

Root/previous symmetry group
- Too restricted

Pointwise stabilizer
- Fixed variables are held in place by the symmetries
- Easy to compute, quick approximation
- Cannot grow

Setwise stabilizer
- Equal variables can be permuted to each other
- Larger than the pointwise stabilizer
- Can grow

Submatrix symmetries
- The largest group
- Slow to compute, leads to duplicate rows
Symmetry and Heuristics

Question
Given a feasible solution and a set of constraint symmetries how to get better feasible solutions?

Orbit mipping
- Fix the number of nonzeros (sum of variables) on each orbit and resolve
- Reaches more than the orbit of the solution
- Potentially slow

Symmetry search/optimize over the group
- Use the symmetries to generate other feasible solutions
- Can get stuck in local optimum
- Very fast
Symmetry – Effect on the Tree

millions of nodes

279 nodes
Symmetry – Results and Plans

Results
- 15% speedup overall on our test set
- Symmetry detection takes less than 1s in most cases
- Warmstart yields 30% speedup for symmetry detection

Plans
- Presolver
  - aggregate variables on an orbit
  - works for IPM, simplex needs crossover
- LP
  - orbit pricing
  - degeneracy
- MILP
  - orbit probing
Parallel Tree Search

Implementation

- Shared-memory threaded computation
- Deterministic synchronization
  » based on quantity and quality of nodes
- Shared bounds, cuts, branching stats, conflict information, ...
- Dynamic reallocation of nodes to threads

Results

- 2x speedup on 4 threads on our test set
Other Improvements

**LP**
- Various crossover improvements

**MILP**
- LP/MILP interface improvements

**DECOMP**
- MILP heuristics on compact formulation
- Identical subproblems, aggregate reformulation
- Ryan-Foster branching for set partitioning

**NLP**
- OPTLSO (local search): multiple objectives, Pareto frontier
- Speed improvements
References


Other SAS/OR talks

SA04 New Approaches to Mixed Integer Programming Heuristics
  Menal Güzelsoy, Philipp Christophel, Imre Pólik

MC08 New Features in OPTMODEL: Parallel Coroutines and Semantic Modeling
  Leo Lopes

MC23 The Value of the Price Envelope
  Bahadir Aral

MC23 Revenue Management as a Strategic Tool
  Maarten Oosten

MD08 Dip and DipPy: A Decomposition-based Modeling System and Solver
  co-author Matthew Galati

MD10 Managing Successful Careers in O.R. Practice - A Panel Discussion
  including Radhika Kulkarni

MD29 Building and Solving Optimization Models with SAS
  Ed Hughes, Rob Pratt

WD23 Multicriteria Model for Price and Promotion Optimization
  Natalia Viktorovna, Maarten Oosten