FilMINT: A Linearizations-based MINLP Solver

Kumar Abhishek[1] Sven Leyffer[2] Jeff Linderoth[3]

[1]ISE Department
Lehigh University
kua3@lehigh.edu

[2]Mathematics and Computer Science Division
Argonne National Laboratory
leyffer@mcs.anl.gov

[3]ISE Department
Lehigh University
jtl3@lehigh.edu

INFORMS Annual Meeting, Pittsburgh.
Nov 07, 2006

MINLP Formulation

$$
\begin{aligned}
z_{\mathrm{MINLP}} = \text{minimize} \quad & f(x, y) \\
\text{subject to} \quad & g_j(x, y) \leq 0, \quad j = 1, \ldots, m, \\
& x \in X, y \in Y \cap \mathbb{Z}^p,
\end{aligned}
\qquad \text{(MINLP)}
$$

where

$$
X \stackrel{\text{def}}{=} \{x \mid x \in \mathbb{R}^n, Dx \leq d\},
$$

$$
Y \stackrel{\text{def}}{=} \{y \mid y \in \mathbb{R}^p, Ay \leq a, y^l \leq y \leq y^u\}.
$$

- f, g_j are twice continuously differentiable (convex) functions.
- x and y are continuous and discrete variables respectively.

- An NP-hard problem.
- A number of interesting applications...

Motivation

- LP/NLP Algorithm by Quesada and Grossmann [1992].
- Early implementation by Leyffer [1993]: $10\times$ faster than OA.
- Advent of modern, flexible Branch and Cut framework.
- Want to use MILP framework's advanced features for our problem.
- Steady improvements in nonlinear programming solvers.
- FilMINT uses FilterSQP, a robust, active set solver for solving NLPs, and MINTO for the MILP framework.
- In order to describe the algorithm, we next define some problems.

NLP subproblem for a fixed y (say y^k)

$$z_{\mathrm{NLP}(y^k)} = \begin{array}{ll} \text{minimize} & f(x, y^k) \\ \text{subject to} & g_j(x, y^k) \le 0 \quad j = 1, \ldots, m, \\ & x \in X. \end{array} \qquad (\mathrm{NLP}(y^k))$$

$\Rightarrow$ Solution is x^k.

- NLP(y^k) feasible $\Rightarrow$ Upper Bound.
- If NLP(y^k) infeasible, NLP solver detects this and gives solution to:

Feasibility subproblem for fixed y^k

$$\begin{array}{ll} \text{minimize} & \displaystyle\sum_{j=1}^{m} w_j g_j(x, y^k)^+, \\ \text{subject to} & x \in X. \end{array} \qquad (\mathrm{NLPF}(y^k))$$

NLP Relaxation for a node with bounds (l, u) on y

$$
\begin{aligned}
z_{\text{NLPR}(l,u)} = \text{minimize} \quad & f(x,y) \\
\text{subject to} \quad & g_j(x,y) \le 0 \quad j = 1, \dots, m, \\
& x \in X, y \in Y, \\
& l \le y \le u.
\end{aligned}
\qquad \text{(NLPR}(l, u))
$$

- Convexity of f and $g_j \Rightarrow$ linearizations about any point (x^k, y^k) outer-approximate the feasible set, and underestimate the objective function.

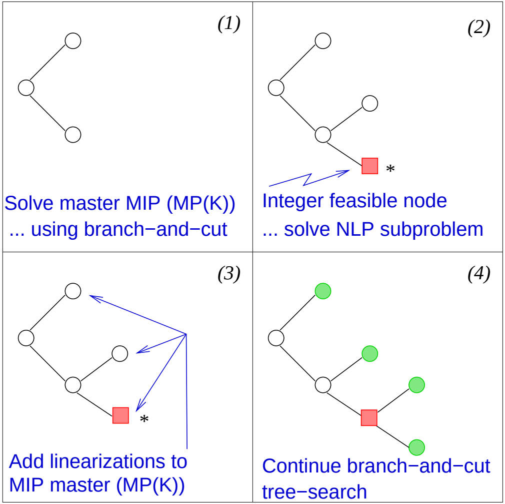
$$
f(x^k,y^k) + \nabla f(x^k,y^k)^T \begin{bmatrix} x - x^k \\ y - y^k \end{bmatrix} \le \eta \qquad \text{(OA}(x_k, y_k))
$$

$$
g_j(x^k,y^k) + \nabla g_j(x^k,y^k)^T \begin{bmatrix} x - x^k \\ y - y^k \end{bmatrix} \le 0 \quad j = 1, \dots, m.
$$

Outer-Approximation based MILP Master Problem

$$
\begin{aligned}
z_{\mathrm{MP}(\mathcal{K})} = \text{minimize} \quad & \eta \\
\text{subject to} \quad & f(x^k,y^k) + \nabla f(x^k,y^k)^T \begin{bmatrix} x - x^k \\ y - y^k \end{bmatrix} \leq \eta \ \ \forall (x^k,y^k) \in \mathcal{K} \qquad \text{(MP}(\mathcal{K})\text{)} \\
& g_j(x^k,y^k) + \nabla g_j(x^k,y^k)^T \begin{bmatrix} x - x^k \\ y - y^k \end{bmatrix} \leq 0 \ \ \forall (x^k,y^k) \in \mathcal{K} \quad j = 1,\ldots,m \\
& x \in X, y \in Y \cap \mathbb{Z}^p.
\end{aligned}
$$

LP/NLP based Branch and Bound Algorithm



Computational Experiments and features explored

- More than 250 MINLP instances from various sources.
- Classified as easy (< 1 min), moderate (1 min - 1 hour), and hard (> 1 hour) using MINLP-BB, a nonlinear branch-and-bound solver.
- Ran with 4 hour time limit on processors with 1.8GHz clockspeed and 2 Gb RAM.
- Show performance profiles to summarize the results.
- Probability that solver i is at most x-times worse than the best.
- Time used as metric for moderate instances, and solution value (gap to best known upper bound) for hard instances.

We now show the features explored with FilMINT:

1. MILP features
2. Linearization management.
3. Linearization generation.

MIP features

- Preprocessing the master problem.
- Cutting planes - use cut generation routines in MINTO.
- Primal heuristic - use the diving based heuristic in MINTO.
- Branching rules
  - Maximal fractional branching.
  - Strong branching.
  - Pseudo-cost based branching.
- Node selection strategies
  - Best bound.
  - Depth first.
  - Best estimate.
  - Adaptive (best bound + best estimate).

Performance Profiles

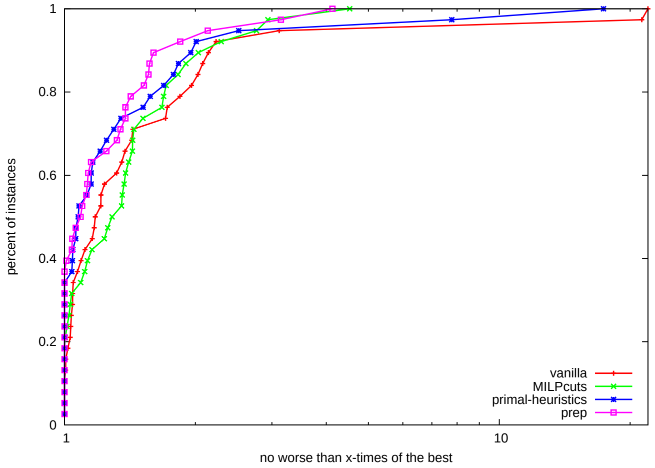


Figure: Effect of cuts, heuristics and preprocessing for moderate instances.

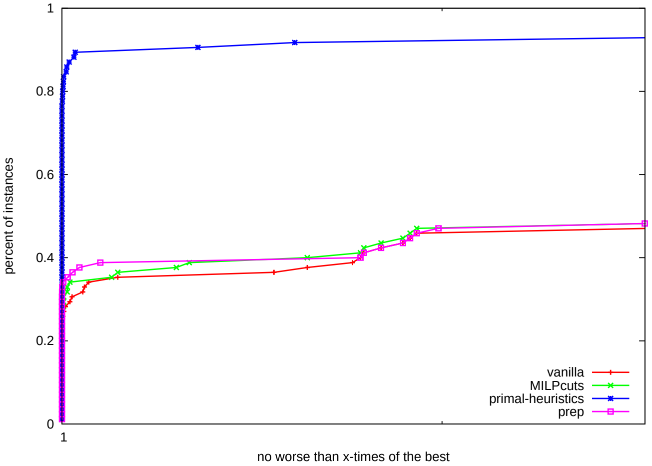


Figure: Effect of cuts, heuristics and preprocessing for hard instances.

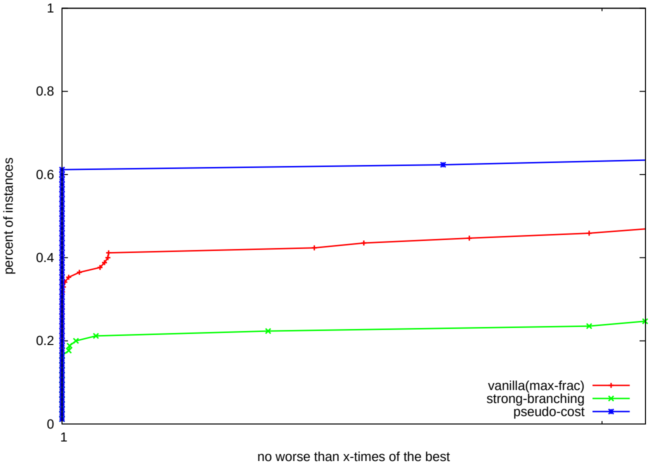


Figure: Effect of branching rules for hard instances.

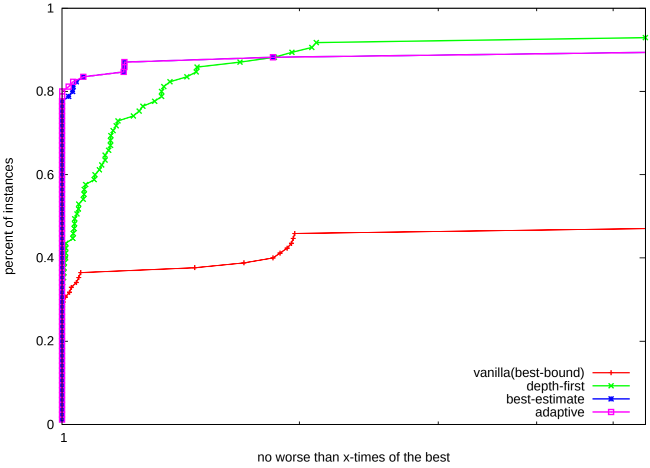


Figure: Effect of node selection rules for hard instances.

Managing linearizations

Large number of linearizations and other inequalities
$\Rightarrow$ increases the LP solve time and memory requirements.

- Add linearizations only if violated by the current LP solution.
- Remove constraints if they remain inactive for a long time.
    - Make use of MINTO's row management scheme to do this.
    - MINTO checks if constraint inactive for last 15 iterations.

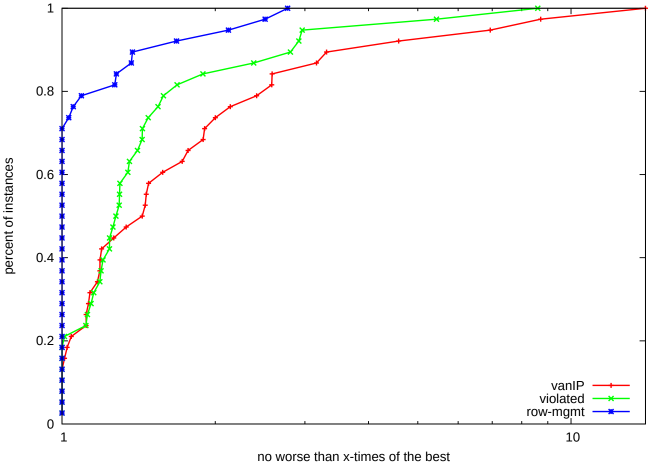


Figure: Effect of linearization management for moderate instances.

Performance Profiles

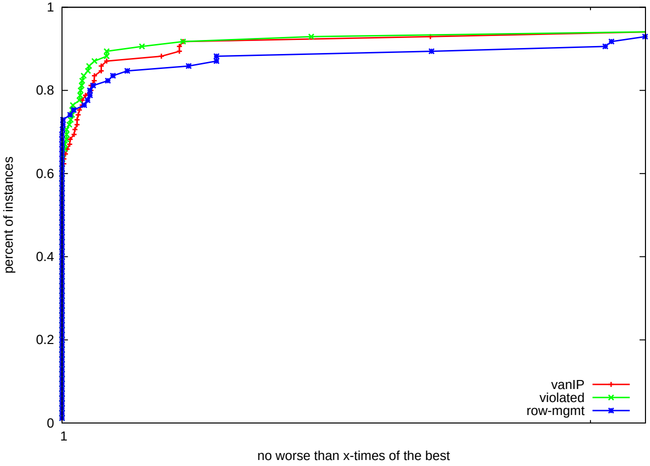
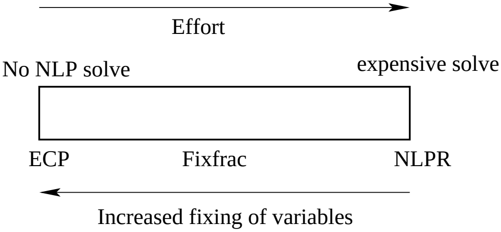


Figure: Effect of linearization management for hard instances.

Generating linearizations

We consider simple extensions to generate new linearizations.

- About which points can we generate linearizations.
- How much work to do to generate them.
- Linearizations from NLP(y^k) fixed at fractional y^k.
  - We call this fixfrac.
  - Do not have to wait for integer feasible solution.
- ECP method: do gradient evaluations and linearize instead of solving NLPs.
- We can solve an NLP by varying the number of variables that we fix.



Cut generation strategy depends on:

- $\rho(d)$ - probability of generation of new linearizations at a node.
  - Depends on the depth d of the node.
  - $\rho(d) \equiv \beta 2^{-d}$.
- At present: same strategy for fixfrac, ECP, and NLPR.

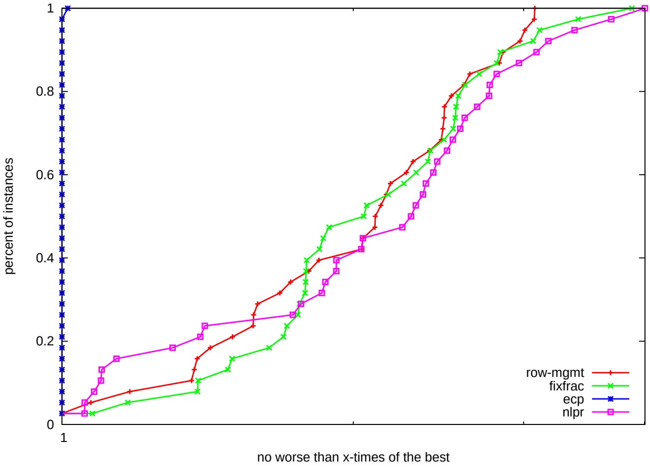


Figure: Effect of linearization generation for moderate instances.

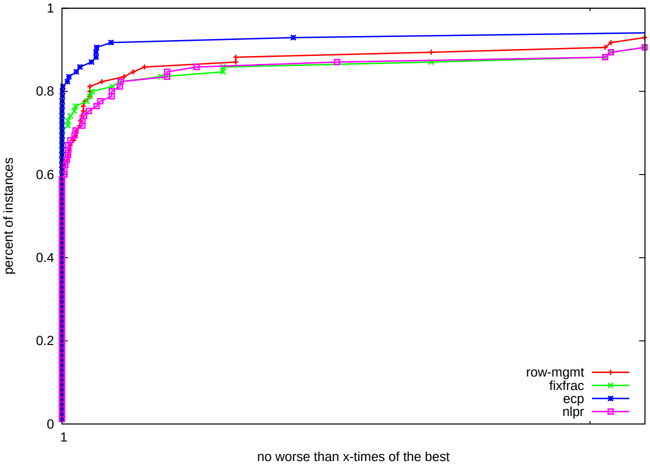


Figure: Effect of linearization generation for hard instances.

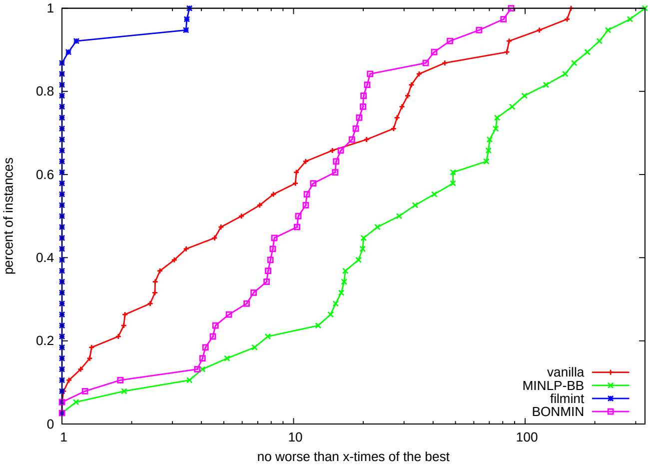


Figure: Comparing FilMINT with other solvers for moderate instances.

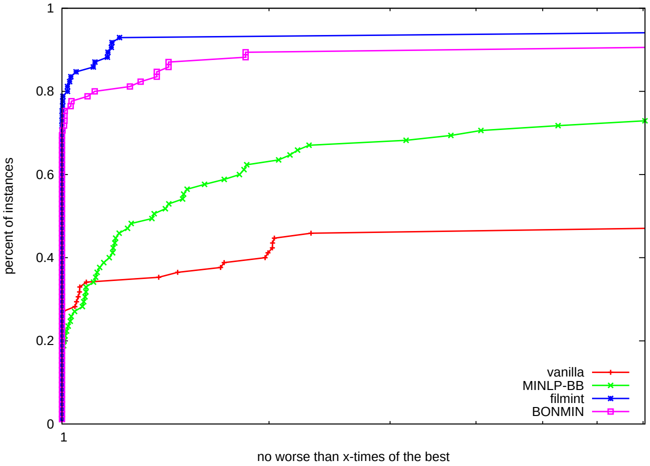


Figure: Comparing FilMINT with other solvers for hard instances.

Conclusions

- Introduce a new solver FilMINT, based on the LP/NLP algorithm in a branch-and-cut framework.
- Create using existing software components.
- Investigate impact of MIP features.
- New ways of generating and managing linearizations.
- FilMINT outperforms other MINLP solvers.
- Future research: cutting plane techniques for MINLP.