A Lifted Linear Programming Branch-and-Bound Algorithm for Mixed Integer Conic Quadratic Programs

Juan Pablo Vielma Shabbir Ahmed
George L. Nemhauser

H. Milton Stewart School of Industrial and Systems Engineering
Georgia Institute of Technology

SIAM Conference on Optimization, 2008 – Boston

Outline

1. Introduction
2. Lifted LP Algorithm
3. Computational Results
4. Final Remarks

"Convex" Mixed Integer Non-Linear Programming (MINLP) Problems

$$
\begin{aligned}
z_{\mathsf{MINLP}} := \max_{x,y} \quad & cx + dy \\
s.t. \quad & (x, y) \in \mathcal{C} \subset \mathbb{R}^{n+p} \\
& x \in \mathbb{Z}^n
\end{aligned}
\qquad \text{(MINLP)}
$$

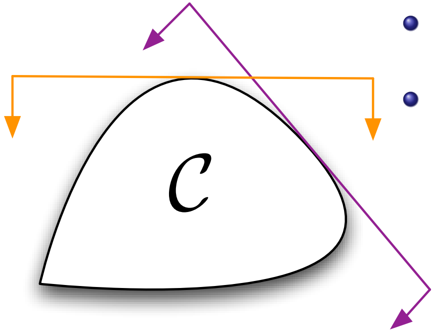
- $\mathcal{C}$ is a convex compact set.
- Advanced algorithms and Software:
  - NLP based branch-and-bound algorithms (Borchers and Mitchell, 1994, Gupta and Ravindran, 1985, Leyffer 2001 and Stubbs and Mehrotra, 1999,. . .)
  - Polyhedral relaxation based algorithms (Duran and Grossmann, 1986, Fletcher and Leyffer, 1994, Geoffrion, 1972,Quesada and Grossmann, 1992, Westerlund and Pettersson, 1995,Westerlund et al., 1994,. . .)
  - CPLEX 9.0+ (ILOG, 2005), Bonmin (Bonami et al., 2005), FilMINT (Abhishek et al., 2006), . . .

"Convex" Mixed Integer Non-Linear Programming (MINLP) Problems

$$
\begin{aligned}
z_{\text{MINLP}} := \max_{x,y} \quad & cx + dy \\
s.t. \quad & (x, y) \in \mathcal{C} \subset \mathbb{R}^{n+p} \\
& x \in \mathbb{Z}^n
\end{aligned}
\qquad \text{(MINLP)}
$$

- $\mathcal{C}$ is a convex compact set.
- Advanced algorithms and Software:
  - NLP based branch-and-bound algorithms (Borchers and Mitchell, 1994, Gupta and Ravindran, 1985, Leyffer 2001 and Stubbs and Mehrotra, 1999,...)
  - Polyhedral relaxation based algorithms (Duran and Grossmann, 1986, Fletcher and Leyffer, 1994, Geoffrion, 1972,Quesada and Grossmann, 1992, Westerlund and Pettersson, 1995,Westerlund et al., 1994,...)
  - CPLEX 9.0+ (ILOG, 2005), Bonmin (Bonami et al., 2005), FilMINT (Abhishek et al., 2006), ...
- Polyhedral relaxation algorithms try to exploit the technology for Mixed Integer Linear Programming

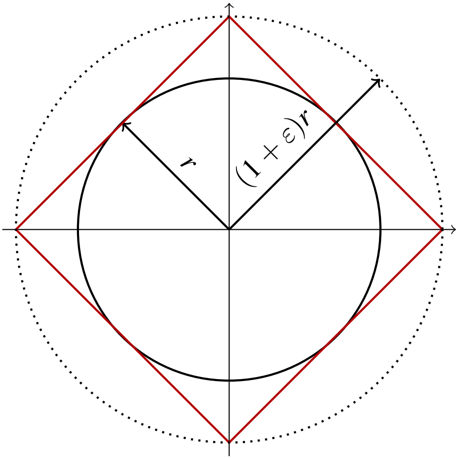
Polyheral Relaxation Based Algorithms



- Approximate convex sets using gradient cuts (tangent, benders).
- Cuts are in the original space.
- Usually only a few cuts are necessary.
- Sometimes convergence of cutting plane procedure is bad (e.g. Quadratic constraints).
  - Solution: Use a polyhedral approximation of the whole set.

Polyheral Relaxation of Convex Sets

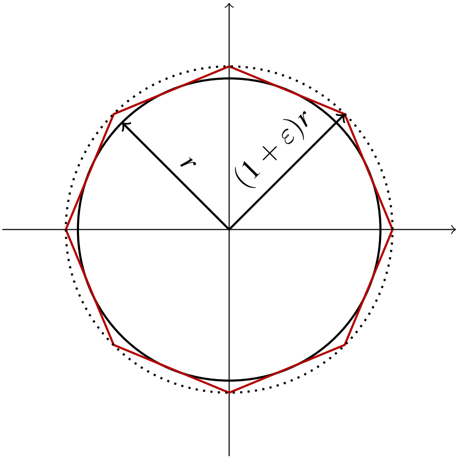
$\mathcal{C} = \mathcal{B}^d(r)$, $d = 2$, $\varepsilon = 0.41$



- It is known that at least $\exp(d/(2(1+\varepsilon))^2)$ facets are needed in the original space.
- Ben-Tal and Nemirovski (2001) approximate $\mathcal{B}^d(r)$ as the projection of a polyhedron with $O(d \log(1/\varepsilon))$ variables and constraints.
- Glineur (2000) refined the approximation and showed that it is algorithmically and computationally "impractical" for (pure continuous) conic quadratic optimization.

Polyheral Relaxation of Convex Sets

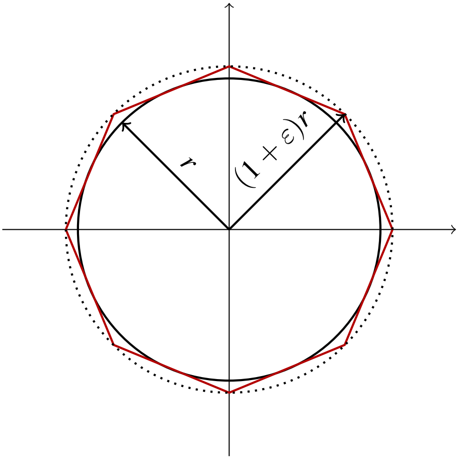
$\mathcal{C} = \mathcal{B}^d(r),\ d = 2,\ \varepsilon = 0.08$



- It is known that at least $\exp(d/(2(1+\varepsilon))^2)$ facets are needed in the original space.
- Ben-Tal and Nemirovski (2001) approximate $\mathcal{B}^d(r)$ as the projection of a polyhedron with $O(d \log(1/\varepsilon))$ variables and constraints.
- Glineur (2000) refined the approximation and showed that it is algorithmically and computationally "impractical" for (pure continuous) conic quadratic optimization.

Polyheral Relaxation of Convex Sets

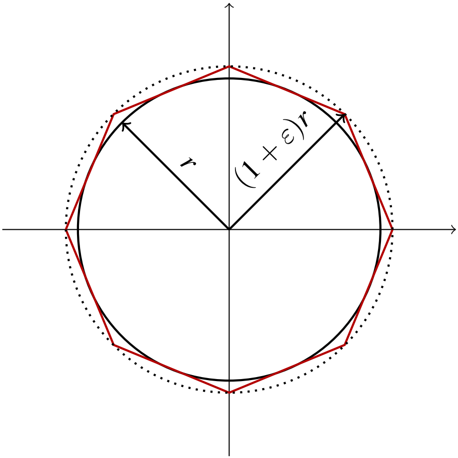
$\mathcal{C} = \mathcal{B}^d(r)$, $d = 2$, $\varepsilon = 0.08$



- It is known that at least $\exp(d/(2(1+\varepsilon))^2)$ facets are needed in the original space.
- Ben-Tal and Nemirovski (2001) approximate $\mathcal{B}^d(r)$ as the projection of a polyhedron with $O(d \log(1/\varepsilon))$ variables and constraints.
- Glineur (2000) refined the approximation and showed that it is algorithmically and computationally “impractical” for (pure continuous) conic quadratic optimization.

Polyheral Relaxation of Convex Sets

$\mathcal{C} = \mathcal{B}^d(r),\ d = 2,\ \varepsilon = 0.08$



- It is known that at least $\exp(d/(2(1+\varepsilon))^2)$ facets are needed in the original space.
- Ben-Tal and Nemirovski (2001) approximate $\mathcal{B}^d(r)$ as the projection of a polyhedron with $O(d\log(1/\varepsilon))$ variables and constraints.
- Glineur (2000) refined the approximation and showed that it is algorithmically and computationally “impractical” for (pure continuous) conic quadratic optimization.

Using Ben-Tal Nemirovski Approximation to Exploit Mixed Integer Linear Programming Solver Technology

- Lifted linear programming relaxation: Polyhedron $\mathcal{P} \subset \mathbb{R}^{n+p+q}$ such that

$$\mathcal{C} \subset \{(x,y) \in \mathbb{R}^{n+p} \ : \ \exists\, v \in \mathbb{R}^{q} \text{ s.t. } (x,y,v) \in \mathcal{P}\} \approx \mathcal{C}$$

- Use a state of the art MILP solver to solve

$$\begin{aligned} \max_{x,y,v} \quad & cx + dy \\ s.t. \quad & (x,y,v) \in \mathcal{P} \\ & x \in \mathbb{Z}^n \end{aligned} \qquad \text{(MILP)}$$

- Problem: Obtained solution might not even be feasible for MINLP
- Solution: Modify Solve of MILP

Idea: Simulate NLP Branch-and-Bound

- Problem solved in NLP B&B node $(l^k, u^k) \in \mathbb{Z}^{2n}$ is:

$$\begin{aligned} z_{\text{NLP}(l^k,u^k)} := \max_{x,y} \quad & cx + dy \\ s.t. \quad & (x,y) \in \mathcal{C} \subset \mathbb{R}^{n+p} \\ & l^k \leq x \leq u^k \end{aligned} \qquad (\text{NLP}(l^k, u^k))$$

- Problem solved by state of the art MILP solver is:

$$\begin{aligned} z_{\text{LP}(l^k,u^k)} := \max_{x,y,v} \quad & cx + dy \\ s.t. \quad & (x,y,v) \in \mathcal{P} \\ & l^k \leq x \leq u^k \end{aligned} \qquad (\text{LP}(l^k, u^k))$$

Idea: Simulate NLP Branch-and-Bound

- Problem solved in NLP B&B node $(l^k, u^k) \in \mathbb{Z}^{2n}$ is:

$$\begin{aligned} z_{\mathrm{NLP}(l^k,u^k)} := \max_{x,y} \quad & cx + dy \\ s.t. \quad & (x,y) \in \mathcal{C} \subset \mathbb{R}^{n+p} \qquad (\mathrm{NLP}(l^k,u^k)) \\ & l^k \le x \le u^k \end{aligned}$$

- Problem solved by state of the art MILP solver is:

$$\begin{aligned} z_{\mathrm{LP}(l^k,u^k)} := \max_{x,y,v} \quad & cx + dy \\ s.t. \quad & (x,y,v) \in \mathcal{P} \qquad (\mathrm{LP}(l^k,u^k)) \\ & l^k \le x \le u^k \end{aligned}$$

- Advantages of second subproblem:
  - Algorithmic Advantage: Simplex has warm starts.
  - Computational Advantage: Use MILP solver's technology.

Idea: Simulate NLP Branch-and-Bound

- Problem solved in NLP B&B node $(l^k, u^k) \in \mathbb{Z}^{2n}$ is:

$$
\begin{aligned}
z_{\text{NLP}(l^k,u^k)} := \max_{x,y} \quad & cx + dy \\
s.t. \quad & (x,y) \in \mathcal{C} \subset \mathbb{R}^{n+p} \qquad \qquad (\text{NLP}(l^k, u^k)) \\
& l^k \le x \le u^k
\end{aligned}
$$

- Problem solved by state of the art MILP solver is:

$$
\begin{aligned}
z_{\text{LP}(l^k,u^k)} := \max_{x,y,v} \quad & cx + dy \\
s.t. \quad & (x,y,v) \in \mathcal{P} \qquad \qquad (\text{LP}(l^k, u^k)) \\
& l^k \le x \le u^k
\end{aligned}
$$

- Issues:
  1. Integer feasible solutions may be infeasible for $\mathcal{C}$.
  2. Need to be careful when fathoming by integrality.

First Issue: Correcting Integer Feasible Solutions

- Let $(x^*, y^*, v^*) \in \mathcal{P}$ such that $x^* \in \mathbb{Z}^n$, but $(x^*, y^*) \notin \mathcal{C}$.
- We reject (x^*, y^*, v^*) and try to correct it using:

$$z_{\text{NLP}(x^*)} := \max_{y} \quad cx^* + dy$$
$$s.t.$$
$$(x^*, y) \in \mathcal{C} \subset \mathbb{R}^{n+p}. \qquad (\text{NLP}(x^*))$$

- This can be done for solutions found by heuristics, at integer feasible nodes, etc.

Second Issue: Correct Fathoming by Integrality

- Suppose that for a node (l^k, u^k) with $l^k \neq u^k$ we have that the solution (x^*, y^*, v^*) of LP(l^k, u^k) is such that $x^* \in \mathbb{Z}^n$
- If $(x^*, y^*) \in \mathcal{C}$ then (x^*, y^*) is also the optimal for NLP(l^k, u^k) and we can fathom by integrality.
- If $(x^*, y^*) \notin \mathcal{C}$ it is not sufficient to solve NLP(x^*):
  - Problem: Corrected solution is not necessarily optimal for NLP(l^k, u^k).
  - Solution: Solve NLP(l^k, u^k) and process node according to its solution.

Second Issue: Correct Fathoming by Integrality

- Suppose that for a node (l^k, u^k) with $l^k \neq u^k$ we have that the solution (x^*, y^*, v^*) of LP(l^k, u^k) is such that $x^* \in \mathbb{Z}^n$
- If $(x^*, y^*) \in \mathcal{C}$ then (x^*, y^*) is also the optimal for NLP(l^k, u^k) and we can fathom by integrality.
- If $(x^*, y^*) \notin \mathcal{C}$ it is not sufficient to solve NLP(x^*):
  - Problem: Corrected solution is not necessarily optimal for NLP(l^k, u^k).
  - Solution: Solve NLP(l^k, u^k) and process node according to its solution.

Second Issue: Correct Fathoming by Integrality

- Suppose that for a node (l^k, u^k) with $l^k \neq u^k$ we have that the solution (x^*, y^*, v^*) of LP(l^k, u^k) is such that $x^* \in \mathbb{Z}^n$
- If $(x^*, y^*) \in \mathcal{C}$ then (x^*, y^*) is also the optimal for NLP(l^k, u^k) and we can fathom by integrality.
- If $(x^*, y^*) \notin \mathcal{C}$ it is not sufficient to solve NLP(x^*):
  - Problem: Corrected solution is not necessarily optimal for NLP(l^k, u^k).
  - Solution: Solve NLP(l^k, u^k) and process node according to its solution.

Computational Experiments

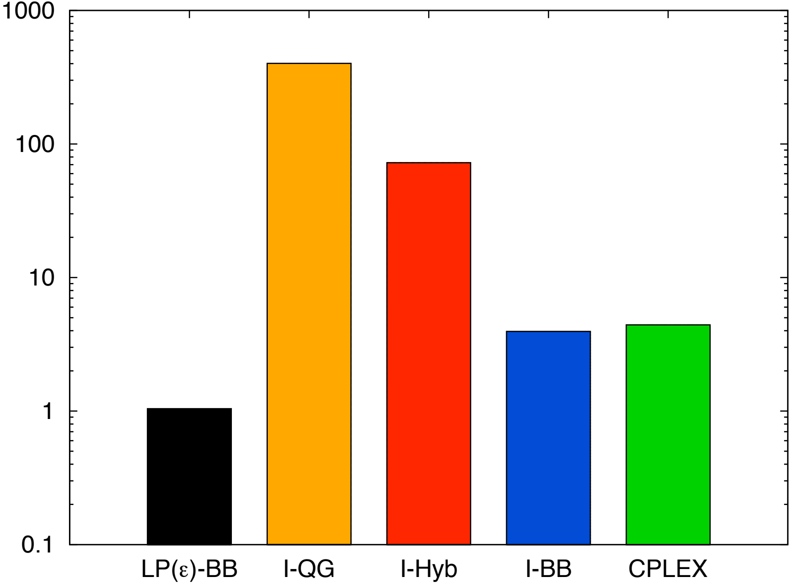
- Implementation of Lifted LP B&B Algorithm (LP(ε) -BB):
  - Using Ben-Tal Nemirovski relaxation from Glineur (2000).
  - Implemented by modifying CPLEX 10's MILP solver using branch, incumbent and heuristic callbacks.
  - $\varepsilon = 0.01$ was selected after calibration experiments.
- Portfolio optimization problems with cardinality constraints (Ceria and Stubbs, 2006; Lobo et al., 1998, 2007):
  - 3 types, all restricting investment in at most 10 stocks.
  - Random selection from S&P 500.
  - 100 instances for $n \in \{20, 30, 40, 50\}$, 10 for $n \in \{100, 200\}$.
- Computer and solvers:
  - Dual 2.4GHz Xeon Linux workstation with 2GB of RAM.
  - LP(ε) -BB v/s CPLEX 10's MIQCP solver and Bonmin's I-BB, I-QG and I-Hyb.

Problem 1: Classical

$$
\begin{aligned}
\max_{x,y} \quad & \bar{a}y \\
s.t. \quad & \\
& ||Q^{1/2}y||_2 \leq \sigma \\
& \sum_{j=1}^{n} y_j = 1 \\
& y_j \leq x_j \qquad \forall j \in \{1, \ldots, n\} \\
& \sum_{j=1}^{n} x_j \leq 10 \\
& x \in \{0,1\}^n \\
& y \in \mathbb{R}^n_+
\end{aligned}
$$

- y fraction of the portfolio invested in each of n assets.
- $\bar{a}$ expected returns of assets.
- $Q^{1/2}$ positive semidefinite square root of the covariance matrix Q of returns.
- Hold at most 10 assets.

Average of Solve Times [s] for $n \in \{20, 30\}$



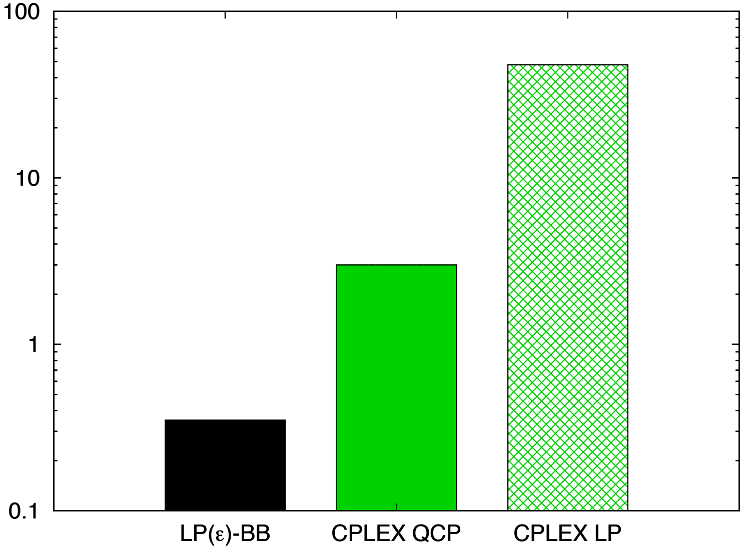
Total Number of Nodes and Calls to Relaxations for Small Instances

I-QG (B&B nodes)	3,580,051
I-Hyb (B&B nodes)	328,316
I-BB (B&B nodes)	68,915
CPLEX (B&B nodes)	85,957
LP(ε) -BB (B&B nodes)	57,933

Total Number of Nodes and Calls to Relaxations for Small Instances

I-QG (B&B nodes)	3,580,051
I-Hyb (B&B nodes)	328,316
I-BB (B&B nodes)	68,915
CPLEX (B&B nodes)	85,957
LP(ε) -BB (B&B nodes)	57,933
NLP(l^k, u^k) (LP(ε) -BB calls)	2,305
NLP(x^*) (LP(ε) -BB calls)	7,810

Avg. of Solve Times [s] for $n \in \{20, 30\}$ (CPLEX v11)



Final Remarks

- Polyhedral relaxation algorithm for “convex” MINLP:
  - Based on a lifted polyhedral relaxation.
  - “Does not update the relaxation“.
- Algorithm for the conic quadratic case:
  - Characteristics:
    - Based on a lifted polyhedral relaxation by Ben-Tal and Nemirovski.
    - Implemented by modifying CPLEX MILP solver.
  - Advantages:
    - Can outperform other methods for portfolio optimization problems.
    - Shows that Ben-Tal and Nemirovski approximation can be computationally “practical”.

Problem 1: Classical

$$
\begin{aligned}
\max_{x,y} \quad & \bar{a}y \\
s.t. \quad & \\
& ||Q^{1/2}y||_2 \leq \sigma \\
& \sum_{j=1}^{n} y_j = 1 \\
& y_j \leq x_j \qquad \forall j \in \{1, \ldots, n\} \\
& \sum_{j=1}^{n} x_j \leq K \\
& x \in \{0,1\}^n \\
& y \in \mathbb{R}^n_+
\end{aligned}
$$

- y fraction of the portfolio invested in each of n assets.
- $\bar{a}$ expected returns of assets.
- $Q^{1/2}$ positive semidefinite square root of the covariance matrix Q of returns.
- K maximum number of assets to hold.

Problem 2 : Shortfall

$$\max_{x,y} \quad \bar{a}y$$

$$s.t.$$

$$||Q^{1/2}y||_2 \leq \sigma$$

$$\sum_{j=1}^{n} y_j = 1$$

$$y_j \leq x_j \qquad \forall j \in \{1,\ldots,n\}$$

$$\sum_{j=1}^{n} x_j \leq K$$

$$x \in \{0,1\}^n$$

$$y \in \mathbb{R}^n_+$$

- y fraction of the portfolio invested in each of n assets.
- $\bar{a}$ expected returns of assets.
- $Q^{1/2}$ positive semidefinite square root of the covariance matrix Q of returns.
- K maximum number of assets to hold.

Problem 2 : Shortfall

$$\max_{x,y} \quad \bar{a}y$$

$$s.t.$$

$$||Q^{1/2}y||_2 \leq \frac{\bar{a}y - W_i^{low}}{\Phi^{-1}(\eta_i)} \qquad i \in \{1, 2\}$$

$$\sum_{j=1}^{n} y_j = 1$$

$$y_j \leq x_j \qquad \forall j \in \{1, \dots, n\}$$

$$\sum_{j=1}^{n} x_j \leq K$$

$$x \in \{0, 1\}^n$$

$$y \in \mathbb{R}_+^n$$

- y fraction of the portfolio invested in each of n assets.
- $\bar{a}$ expected returns of assets.
- $Q^{1/2}$ positive semidefinite square root of the covariance matrix Q of returns.
- K maximum number of assets to hold.
- Approximation of $\text{Prob}(\bar{a}y \geq W_i^{low}) \geq \eta_i$

Problem 3 : Robust

$$\max_{x,y,r} \quad r$$

$$s.t.$$

$$||Q^{1/2}y||_2 \leq \sigma$$

$$\alpha||R^{1/2}y||_2 \leq \bar{a}y - r$$

$$\sum_{j=1}^{n} y_j = 1$$

$$y_j \leq x_j \qquad \forall j \in \{1, \ldots, n\}$$

$$\sum_{j=1}^{n} x_j \leq K$$

$$x \in \{0,1\}^n$$

$$y \in \mathbb{R}_+^n$$

- y fraction of the portfolio invested in each of n assets.
- $\bar{a}$ expected returns of assets.
- $Q^{1/2}$ positive semidefinite square root of the covariance matrix Q of returns.
- K maximum number of assets to hold.
- Robust version from uncertainty in $\bar{a}$.

Instance Data

- Maximum number of stocks $K = 10$.
- Maximum risk $\sigma = 0.2$.
- Shortfall constraints: $\eta_1 = 80\%$, $W_1^{low} = 0.9$, $\eta_2 = 97\%$, $W_2^{low} = 0.7$ (Lobo et al., 1998, 2007).
- Data generation for Classical and Shortfall from S&P 500 data following Lobo et al. (1998), (2007).
- Data generation for Robust from S&P 500 data following Ceria and Stubbs (2006).
- Riskless asset included for Shortfall.
- Random selection of n stocks out of 462.
- 100 instances for $n \in \{20, 30, 40, 50\}$, 10 for $n \in \{100, 200\}$.

Branch-and-Bound Main Loop

Set global lower bound $\mathrm{LB} := -\infty$.
Set $l_i^0 := -\infty$, $u_i^0 := +\infty$ for all $i \in \{1, \ldots, n\}$.
Set node list $\mathcal{H} := \{(l^0, u^0)\}$.
while $\mathcal{H} \neq \emptyset$ **do**
    Select and remove a node $(l^k, u^k) \in \mathcal{H}$.
    `ProcessNode`(l^k, u^k).
end

$(\mathrm{LB}, \mathcal{H}) := \texttt{ProcessNode}(l^k, u^k, \mathrm{LB}, \mathcal{H})$

Solve LP(l^k, u^k) (Let (x^*, y^*) be the optimal solution).

if LP(l^k, u^k) *is feasible* **and** $z_{\text{LP}(l^k,u^k)} > \mathrm{LB}$ **then**

  if $x^* \in \mathbb{Z}^n$ **then**

    Solve NLP(x^*).

    if NLP(x^*) *is feasible* **and** $z_{\text{NLP}(x^*)} > \mathrm{LB}$ **then**

      Update LB to $z_{\text{NLP}(x^*)}$.

    end

    Extra Steps

  else

    Branch on x^* and add nodes to $\mathcal{H}$.

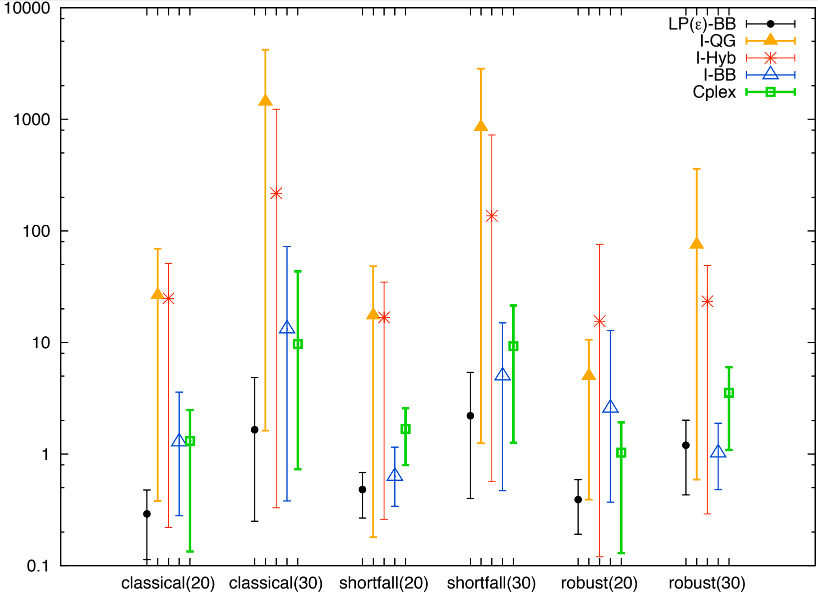
  end

end

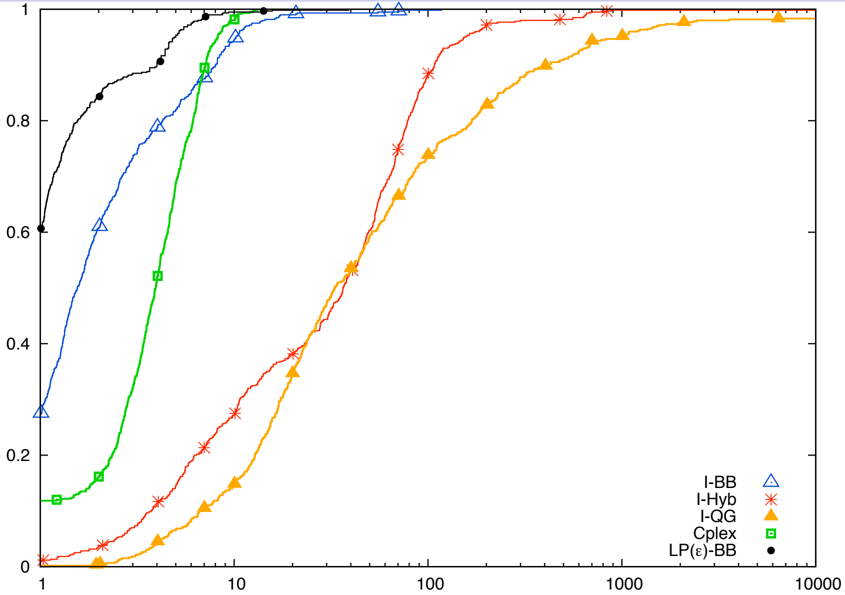
$(\mathrm{LB}, \mathcal{H}) := \texttt{ProcessNode}(l^k, u^k, \mathrm{LB}, \mathcal{H})$

if $l^k \neq u^k$ **then**
  Solve NLP(l^k, u^k) (Let $(\tilde{x}, \tilde{y})$ be the optimal solution).
  if NLP(l^k, u^k) *is feasible* **and** $z_{\text{NLP}(l^k,u^k)} > \mathrm{LB}$ **then**
    if $\tilde{x} \in \mathbb{Z}^n$ **then**
      Update LB to $z_{\text{NLP}(l^k,u^k)}$.
    else
      Branch on $\tilde{x}$ and add nodes to $\mathcal{H}$.
    end
  end
end

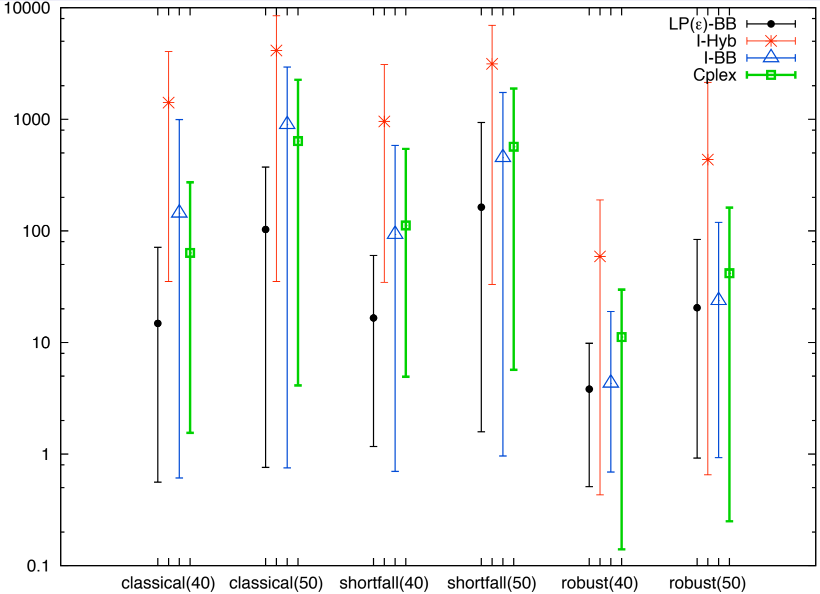
Average Solve Times [s] for $n \in \{20, 30\}$



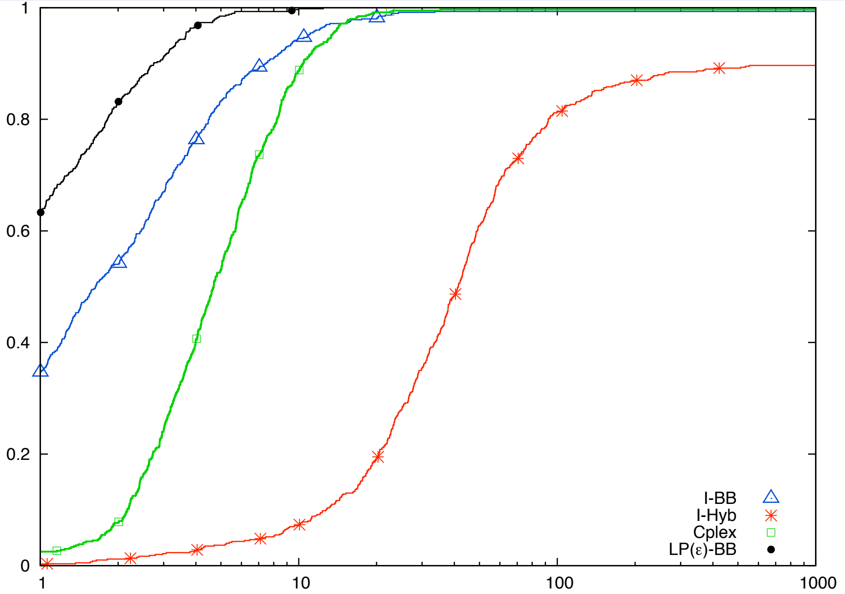
Performance Profile for $n \in \{20, 30\}$



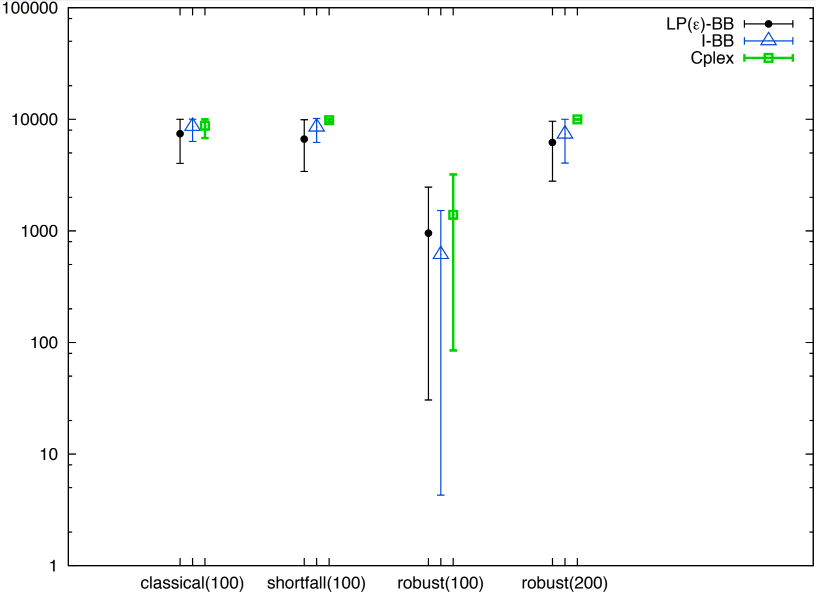
Average Solve Times [s] for $n \in \{40, 50\}$



Performance Profile for $n \in \{40, 50\}$



Average Solve Times [s] for $n \in \{100, 200\}$



Performance Profile for $n \in \{100, 200\}$