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

Juan Pablo Vielma[1]   Shabbir Ahmed[1]   George L. Nemhauser[1]

[1]School of Industrial and Systems Engineering
Georgia Institute of Technology

CRM, Université de Montréal, July-August, 2007

Mixed Integer Non-Linear Programming (MINLP) Problems

$$z_{\text{MINLPP}} := \max_{x,y} \quad cx + dy$$

$$s.t.$$

$$(x, y) \in \mathcal{C} \subset \mathbb{R}^{n+p} \qquad \text{(MINLPP)}$$

$$x \in \mathbb{Z}^n$$

- $\mathcal{C}$ is a convex compact set.
- Assume for simplicity that MINLPP is feasible.
- Also let NLPP be the nonlinear continuous relaxation obtained by eliminating $x \in \mathbb{Z}^n$.

Two Algorithm Approaches for MINLP

- ▶ Non-linear programming (NLP) based branch-and-bound algorithms (Borchers and Mitchell, 1994; Gupta and Ravindran, 1985, Leyffer 2001 and Stubbs and Mehrotra, 1999):
  - ▶ Analog of LP branch-and-bound for MILP.
  - ▶ Implementations: CPLEX 9.0 and 10.0 (ILOG, 2005) and I-BB solver in Bonmin (Bonami et al., 2005)
- ▶ Polyhedral relaxation based algorithms:
  - ▶ Outer approximation (Duran and Grossmann, 1986; Fletcher and Leyffer, 1994)
  - ▶ Generalized Benders decomposition (Geoffrion, 1972).
  - ▶ LP/NLP-based branch-and-bound (Quesada and Grossmann, 1992).
  - ▶ Extended cutting plane method (Westerlund and Pettersson, 1995;Westerlund et al., 1994).
  - ▶ Implementations: I-OA, I-QG and I-Hyb solvers in Bonmin, MINLP solver FilMINT (Abhishek et al., 2006).

Branch-and-Bound Methods

- A branch-and-bound node is defined by $(l^k, u^k) \in \mathbb{Z}^{2n}$.
- The problem solved in a branch-and-bound node (l^k, u^k) is obtained by adding $l^k \leq x \leq u^k$ to some continuous relaxation of MINLPP.
- For example, NLP based branch-and-bound algorithms use

$$z_{\mathsf{NLPP}(l^k, u^k)} := \max_{x, y} \quad cx + dy$$

$$s.t.$$

$$(x, y) \in \mathcal{C} \subset \mathbb{R}^{n+p} \qquad (\mathsf{NLPP}(l^k, u^k))$$
$$x \geq l^k$$
$$x \leq u^k$$

- Polyhedral relaxation based branch-and-bound algorithms use several polyhedral relaxations of $\mathcal{C}$.

Polyheral Relaxation of Convex Sets

- ▶ Bad news: Even approximating the d dimensional unit euclidean ball $\mathcal{B}^d(1)$ is hard. Any $\mathcal{P} \subset \mathbb{R}^d$ such that

$$\mathcal{B}^d(1) \subset \mathcal{P} \subset (1 + \varepsilon)\mathcal{B}^d(1)$$

has at least $e^{(d/(2(1+\varepsilon))^2)}$ facets.

- ▶ To try to resolve this issue, current polyhedral based algorithms generate the relaxations dynamically.

- ▶ Possible solution: Projection of $\mathcal{P} \subset \mathbb{R}^{d+q}$ to $\mathbb{R}^d$ can have an exponential (w/r to facets and variables of $\mathcal{P}$) number of facets.

  - ▶ Exploiting this Ben-Tal and Nemirovski (Ben-Tal and Nemirovski, 2001) gave a relaxation of $\mathcal{B}^d(1)$ with $O(d \log(1/\varepsilon))$ facets and extra variables.
  - ▶ Higher dimensional or *lifted* polyhedral relaxation of convex sets.

Using a fixed lifted polyhedral relaxation

- ▶ Lifted Linear Programming Relaxation of MINLPP:
  - ▶ Polyhedron $\mathcal{P} \subset \mathbb{R}^{n+p+q}$ such that:

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

  - ▶ We get the relaxation of MINLPP (and NLPP):
    $$z_{\mathsf{LPP}} := \max_{x,y,v} \quad cx + dy$$
    $$s.t.$$
    $$(x, y, v) \in \mathcal{P} \qquad \text{(LPP)}$$

- ▶ Basic lifted LP branch-and-bound algorithm:
  - ▶ Branch-and-bound algorithm that solves $\mathsf{LPP}(l^k, u^k)$ in each node (l^k, u^k) trying to mimic NLP based branch-and-bound.
  - ▶ If $\mathcal{P}$ is not tight enough we may have two problems:
    1. Integer feasible solutions might not be in $\mathcal{C}$.
    2. Fathoming by integrality might fathom optimal solution.

Correcting the basic algorithm

1. Let $(x^*, y^*) \in (\mathcal{P} \setminus \mathcal{C}) \cap (\mathbb{Z}^n \times \mathbb{R}^p)$. We correct it to be feasible for $\mathcal{C}$ using:

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

2. If $l^k \neq u^k$ and solution x^* to $\mathsf{LPP}(l^k, u^k)$ is integer feasible we could have a problem:
   - Neither x^* (if $x^* \in \mathcal{C}$) nor the optimal solution to $\mathsf{NLPP}(x^*)$ are necessarily the optimal solutions to $\mathsf{NLPP}(l^k, u^k)$ (The problem we should really be solving in node (l^k, u^k))
   - Solution: Solve $\mathsf{NLPP}(l^k, u^k)$ and hope this doesn't happen too often.
   - This can cause to effectively branch on an integer feasible solution.

Mixed Integer Conic Quadratic Programming Problems

$$z_{\mathsf{MICPP}} := \max_{x,y} \quad cx + dy$$

$$s.t.$$

$$\begin{aligned}
Dx + Ey &\leq f \\
(x, y) &\in \mathcal{CC}_i \quad i \in \mathcal{I} \qquad \text{(MICPP)} \\
(x, y) &\in \mathbb{R}^{n+p} \\
x &\in \mathbb{Z}^n
\end{aligned}$$

▶ $\mathcal{CC}_i$ is a conic quadratic constraint of the form

$$\mathcal{CC} := \{(x, y) \in \mathbb{R}^{n+p} \; : \; ||Ax + By + \delta||_2 \leq ax + by + \delta_0\}$$

▶ Also let CPP be the continuous relaxation obtained by eliminating $x \in \mathbb{Z}^n$.

Ben-Tal Nemirovski Polyhedral Relaxation of CPP

$$z_{\mathsf{LPP}(\varepsilon)} := \max_{x,y,v} \quad cx + dy$$

$$s.t.$$

$$Dx + Ey \leq f \qquad \qquad (\mathsf{LPP}(\varepsilon))$$
$$(x, y, v) \in \mathcal{P}(\mathcal{CC}_i, \varepsilon) \quad i \in \mathcal{I}$$
$$(x, y, v) \in \mathbb{R}^{n+p+q},$$

- $\mathcal{P}(\mathcal{CC}_i, \varepsilon)$ polyhedron with $O((n + p) \log(1/\varepsilon))$ variables and constraints.

Implementation of Lifted LP Branch-and-Bound Algorithm for Mixed Integer Conic Quadratic Programming Problems

- Denoted by $\text{LPP}(\varepsilon)$-BB .
- Using $\text{LPP} = \text{LPP}(\varepsilon)$ and $\text{NLPP} = \text{CPP}$.
- Using a version of the Ben-Tal Nemirovski relaxation introduced by Glineur.
- Implemented by modifying CPLEX 10's MILP solver.
- Using C++, Ilog Concert Technology. Branch, incumbent and heuristic callbacks.
- $\varepsilon = 0.01$ was selected after calibration experiments.

Computational Experiments

- ▶ Dual 2.4GHz Xeon workstation with 2GB of RAM running Linux Kernel 2.4.
- ▶ LPP(ε)-BB v/s CPLEX 10's MIQCP solver and Bonmin's I-BB, I-QG and I-Hyb.
- ▶ Test set:
  - ▶ Three types of portfolio optimization problems with cardinality constraints and n stocks (Ceria and Stubbs, 2006; Lobo et al., 1998, 2007).
  - ▶ 100 instances for each $n \in \{20, 30, 40, 50\}$.
  - ▶ 10 instances foe each $n \in \{100, 200\}$.

Problem 1: Classical

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

$$s.t.$$

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

$$i \in \{1, 2\}$$

- ► y fraction of the portfolio invested in each of n assets.
- ► $\bar{a}$ expected returns of assets.

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

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

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

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

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

- ► $Q^{1/2}$ positive semidefinite square root of the covariance matrix Q of returns.
- ► K maximum number of assets to hold.

Problem 2 : Shortfall

$s.t.$

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

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

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

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

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

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

- 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.
- Approximation of $\text{Prob}(\bar{a}y \geq W_i^{low}) \geq \eta_i$

Problem 3 : Robust

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

$$s.t.$$

$$i \in \{1, 2\}$$

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

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

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

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

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

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

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

- ▶ 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(n)	stat	LPP(ε) -BB	I-QG	I-Hyb	I-BB	CPLEX
classical(20)	min	0.08	0.38	0.22	0.28	0.02
	avg	**0.29**	**26.41**	**24.84**	**1.28**	**1.31**
	max	1.06	222.19	164.71	13.33	7.95
	std	0.18	42.92	26.37	2.31	1.17
classical(30)	min	0.25	1.62	0.33	0.38	0.73
	avg	**1.65**	**1434.86**	**217.25**	**13.19**	**9.68**
	max	27.0	10005.2	10003.3	573.97	324.63
	std	3.21	2768.34	1016.68	59.17	33.68
shortfall(20)	min	0.19	0.18	0.26	0.34	0.03
	avg	**0.48**	**17.42**	**16.78**	**0.63**	**1.68**
	max	1.65	174.62	58.45	3.52	5.19
	std	0.21	30.77	17.96	0.52	0.89
shortfall(30)	min	0.4	1.25	0.57	0.47	1.26
	avg	**2.20**	**847.63**	**136.39**	**5.00**	**9.26**
	max	29.34	10003.1	5907.32	73.81	80.36
	std	3.21	1992.86	588.61	10.00	12.20
robust(20)	min	0.19	0.39	0.12	0.37	0.03
	avg	**0.39**	**4.99**	**15.51**	**2.57**	**1.03**
	max	1.05	33.85	599.46	57.22	3.5
	std	0.20	5.60	60.37	10.25	0.90
robust(30)	min	0.43	0.59	0.29	0.48	0.07
	avg	**1.20**	**75.07**	**23.43**	**1.02**	**3.54**
	max	4.72	2071.47	134.08	4.92	10.76
	std	0.81	284.39	25.49	0.87	2.45

Table: Solve Times for 20 and 30 stocks [s]

Performance Profile for solve times for $n = 40, 50$
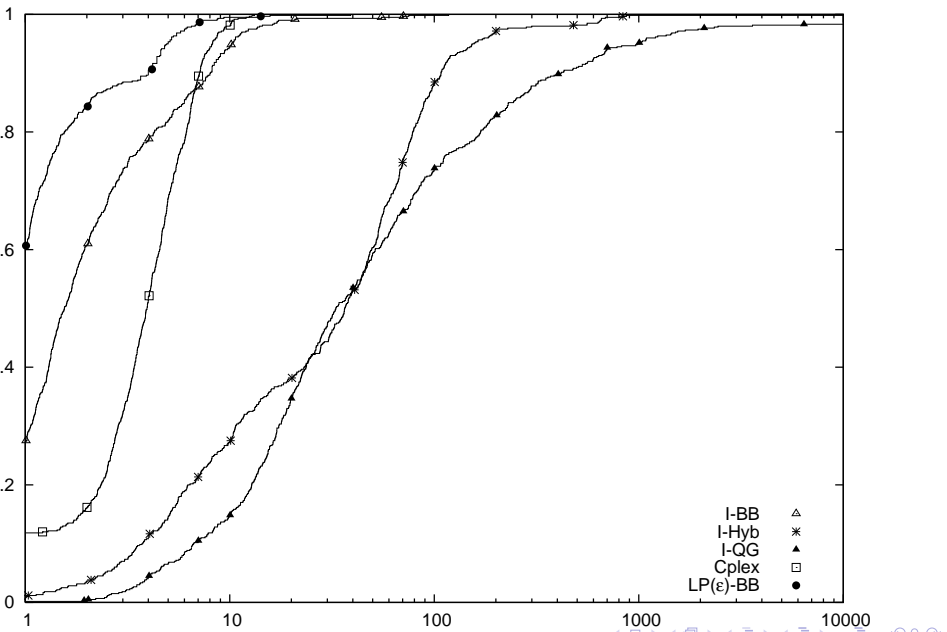
instance(n)	stat	LPP(ε) -BB	I-Hyb	I-BB	CPLEX
classical(40)	min	0.56	35.04	0.61	1.55
	avg	**14.84**	**1412.23**	**144.17**	**63.41**
	max	554.52	10006.0	8518.95	2033.65
	std	56.64	2631.92	848.84	208.86
classical(50)	min	0.76	35.17	0.75	4.12
	avg	**102.88**	**4139.92**	**894.00**	**636.83**
	max	1950.81	12577.8	10030.1	10000.0
	std	270.96	4343.71	2048.96	1626.37
shortfall(40)	min	1.17	34.72	0.7	4.93
	avg	**16.60**	**956.98**	**92.85**	**111.97**
	max	389.57	10004.6	4888.26	4259.5
	std	43.85	2133.56	489.98	430.95
shortfall(50)	min	1.58	33.22	0.96	5.69
	avg	**163.10**	**3143.84**	**452.05**	**567.74**
	max	7674.86	10006.0	10034.1	10000.0
	std	771.98	3803.14	1285.52	1319.39
robust(40)	min	0.51	0.43	0.69	0.14
	avg	**3.82**	**59.10**	**4.31**	**11.17**
	max	42.57	1141.91	129.82	160.71
	std	6.04	130.37	14.64	18.58
robust(50)	min	0.92	0.65	0.93	0.25
	avg	**20.44**	**435.43**	**23.67**	**41.71**
	max	443.29	10002.1	746.37	876.31
	std	63.47	1702.15	95.68	120.24

Table: Solve Times for 40 and 50 stocks [s]

Performance profile for solve times for $n = 40, 50$
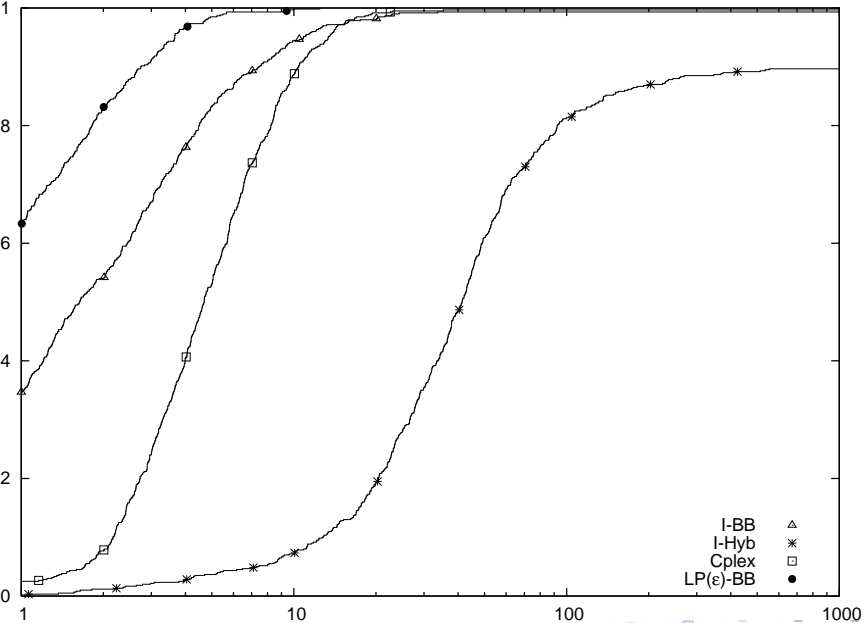
instance(n)	stat	LPP(ε) -BB	I-BB	CPLEX
classical(100)	min	1653	3497	4503
	avg	**7443**	**8605**	**8767**
	max	10012	10035	10000
	solved	4	3	3
shortfall(100)	min	2014	4105	8733
	avg	**6660**	**8497**	**9818**
	max	10003	10163	10000
	solved	6	4	2
robust(100)	min	30	4	85
	avg	**956**	**612**	**1395**
	max	4943	2684	5294
	solved	10	10	10
robust(200)	min	1458	1775	9789
	avg	**6207**	**7346**	**9979**
	max	10138	10016	10000
	solved	6	5	1

Table: Solve Times for 100 and 200 stocks [s]

Performance profile for solve times for $n = 100, 200$
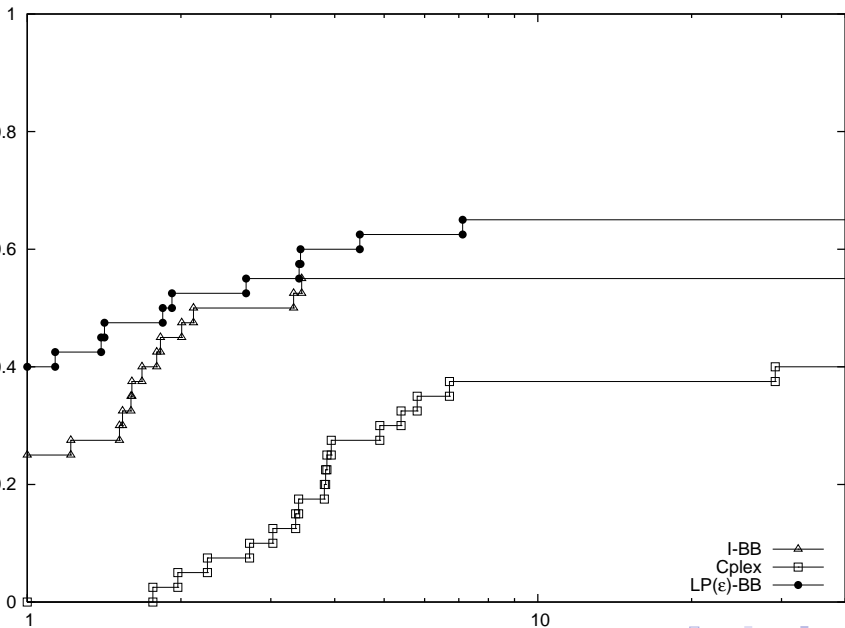
Reasons for the good performance: Lifted LP Relaxation (10 first instances for $n \in \{20, 30\}$)

X Confirming (Glineur, 2000) relaxation is slower than NLP:

	CPP	LPP(0.01)		
stat	Barrier	P. Simplex	D. Simplex	Barrier
min	0.09	0.72	0.58	0.19
avg	**4.88**	**34.94**	**11.66**	**6.94**
max	20.39	174.45	49.69	29.7
std	1.50	10.45	2.86	2.05

Table: Solve Time for Root Relaxation [s].

$\sqrt{}$ Accuracy of the lifted LP relaxation is good:

stat	$\varepsilon = 1$	$\varepsilon = 0.1$	$\varepsilon = 0.01$	$\varepsilon = 0.001$	$\varepsilon = 0.0001$
min	0.25	0.07	0.00	0.00	0.00
avg	**4.30**	**1.14**	**0.07**	**0.00**	**0.00**
max	13.94	5.16	0.29	0.02	0.01
std	0.80	0.21	0.01	0.00	0.00

Table: Accuracy of Relaxation $z_{\text{LPP}(\varepsilon)}$ [%]

Reasons for the good performance: Total Number of Nodes and Calls to Relaxations (All instance for $n \in \{20, 30\}$)

- Very few calls to non-linear relaxations.
- As expected: Fewer nodes than other polyhedral relaxations.
- Somewhat unexpected: Fewer nodes than NLP solvers.

B-and-b nodes I-QG	3580051
B-and-b nodes I-Hyb	328316
B-and-b nodes I-BB	68915
B-and-b nodes CPLEX	85957
B-and-b nodes LPP(ε)-BB	57933
LPP(ε)-BB calls to CPP(l^k, u^k)	2305
LPP(ε)-BB calls to CPP(x^*)	7810

Final Remarks

- Polyhedral relaxation algorithm for MINLP:
  - Based on a lifted polyhedral relaxation.
  - Branches on integer feasible solutions.
  - "Does not update the relaxation".
- Algorithm for the conic quadratic case:
  - Based on a lifted polyhedral relaxation by Ben-Tal and Nemirovski.
  - Implemented by modifying CPLEX MILP solver.
  - Significantly outperforms other methods for portfolio optimization problems.