Computational Approaches to Mixed Integer Second Order Cone Optimization (MISOCO)

Aykut Bulut[1]
Ted Ralphs[1]

[1]COR@L Lab, Department of Industrial and Systems Engineering, Lehigh University

International Symposium on Mathematical Programming,
17 July 2015

Outline

MISOCO definition

- We are interested in solving Mixed Integer Second Order Conic Optimization (MISOCO) problems.
- MISOCO is a generalization of Mixed Integer Linear Optimization (MILP).
- MISOCO can be formulated as follows,

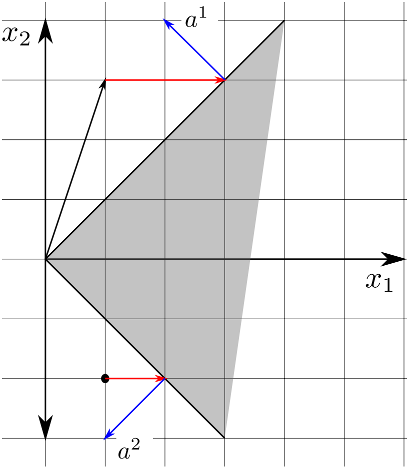
$$\begin{aligned} \min \quad & c^\top x \\ s.t. \quad & Ax = b \\ & x \in \mathbb{L}^1 \times \cdots \times \mathbb{L}^k \\ & x_i \in \mathbb{R}_+ \qquad i \in I \\ & x_j \in \mathbb{Z}_+ \qquad j \in J. \end{aligned} \tag{MISOCO}$$

Outer Approximation Algorithm for SOCO

```
Solve linear relaxation (LP) of the problem.
if LP is infeasible then
    SOCO is infeasible, STOP.
end if
if LP is unbounded then
    while LP is unbounded do
        Determine direction of unboundedness
        if Direction is feasible for all conic constraints then
            SOCO is unbounded, STOP.
        else
            Add cuts using direction of unboundedness.
        end if
        Solve LP.
    end while
end if
Get LP solution
while Solution is not feasbile for conic constraints do
    Add cuts using solution.
    Solve LP.
    if LP is infeasible then
        SOCO is infeasible, STOP.
    end if
    get LP solution.
end while
LP solution is optimal for SOCO, STOP.
```

Separating Infeasible Directions/Solutions

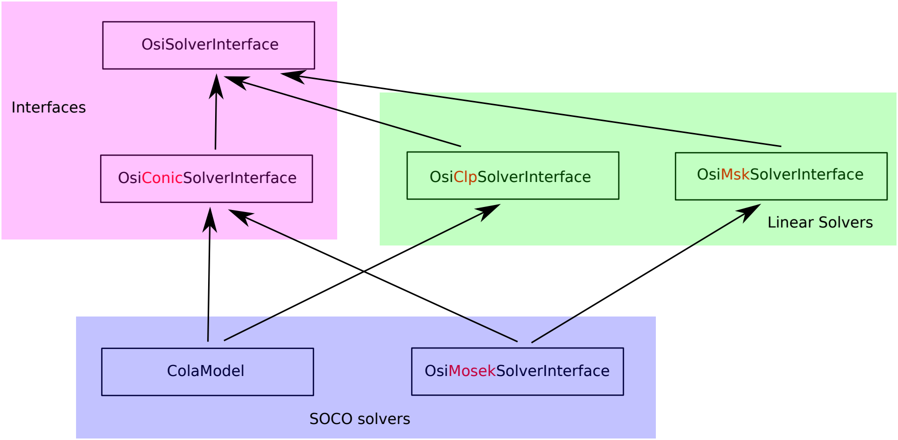
Figure: Separation Example



COLA

- Implements outer approximation algorithm.
- All-written in `C++` language.
- Implements *conic* OSI, which is an extension of OSI.
- Cola uses CLP to solve LP relaxations.
- Reads problems in Mosek's extended MPS format, uses COIN Utils for this.
- *Conic* OSI can be used to build models.
- Takes advantage of simplex method's warm-start capabilities.

Figure: COLA's relationship to COIN-OR projects



COLA Performance Statistics

- NC: Number of Conic constraints the instance has.
- LC: Size of Largest Conic constraint.
- SS: Separation Supports generated.
- MSS: Maximum number of Separation Supports generated for a cone.
- NLP: Number of linear optimization problems solved.
- CPU: CPU seconds spent during execution of COLA.

COLA performance on CBLIB problems 1

Table: COLA statistics on CBLIB 2014 Part 1

instance	NC	LC	SS	MSS	NLP	CPU
chainsing-1000-1	2994	3	14479	10	11	13.01
classical_200_1	1	201	1055	1055	1056	114.11
classical_50_1	1	51	328	328	329	1.89
estein4_A	9	3	36	6	7	0.01
estein4_B	9	3	44	6	9	0.02
estein4_C	9	3	60	10	11	0.02
estein4_nr22	9	3	41	6	7	0.0
estein5_A	18	3	109	11	14	0.02
estein5_nr21	18	3	99	9	11	0.03
pp-n1000-d10000	1000	3	16107	18	19	7.19
pp-n100-d10000	100	3	1613	18	19	0.12
pp-n10-d10000	10	3	161	17	18	0.02
robust_50_1	2	52	260	134	135	0.78
robust_100_1	2	102	577	297	298	7.64
robust_200_1	2	202	960	499	500	64.86
shortfall_100_1	2	101	533	502	503	11.44
shortfall_100_2	2	101	674	630	631	19.28
shortfall_100_3	2	101	573	527	528	12.55
shortfall_200_1	2	201	719	690	691	53.67
shortfall_200_2	2	201	876	841	842	77.74
shortfall_50_1	2	51	307	284	285	1.73
shortfall_50_2	2	51	344	320	321	2.13
shortfall_50_3	2	51	451	408	409	3.58

COLA performance on CBLIB problems 2

Table: COLA statistics on CBLIB 2014 Part 2

instance	NC	LC	SS	MSS	NLP	CPU
sssd-strong-25-8	24	3	243	13	15	0.07
sssd-strong-30-8	24	3	260	13	14	0.07
sssd-weak-20-8	24	3	171	8	9	0.03
sssd-weak-25-8	24	3	171	8	9	0.04
sssd-weak-30-8	24	3	165	8	9	0.03
turbine07_aniso	25	3	53	9	11	0.01
turbine07GF	25	3	10	4	5	0.0
turbine07_lowb_aniso	25	3	64	10	12	0.03
turbine07_lowb	27	9	81	8	9	0.02
turbine07	26	9	67	12	14	0.02
turbine54GF	119	3	25	10	11	0.05
turbine54	120	9	220	11	13	0.05
uflquad-nopsc-10-150	1500	3	14281	16	20	14.68
uflquad-nopsc-20-150	3000	3	29063	17	30	74.84
uflquad-nopsc-30-100	3000	3	29108	23	39	66.91
uflquad-nopsc-30-150	4500	3	42809	19	39	156.26
uflquad-nopsc-30-200	6000	3	55650	19	40	332.71
uflquad-nopsc-30-300	9000	3	83624	16	41	819.0
uflquad-psc-10-150	1500	3	10837	13	23	14.08
uflquad-psc-20-150	3000	3	18164	15	37	70.09
uflquad-psc-30-100	3000	3	16595	22	49	69.07
uflquad-psc-30-150	4500	3	23675	19	50	128.58
uflquad-psc-30-200	6000	3	33972	19	50	291.65
uflquad-psc-30-300	9000	3	54083	19	50	978.33

DisCO solver

- A branch and bound framework to solve MISOCO.
- Uses *conic* OSI to manipulate relaxation problems.
- By default it uses COLA to solve relaxations.
- Cplex and Mosek can also be used through conic OSI interface.
- Extends COIN-OR's High-Performance Parallel Search (CHiPPS) framework for conic problems.
- Similar design to Góez's ICLOPS (developed in his PhD work), major difference is it uses conic OSI and COLA.
- Simplex is used when COLA is chosen as solver.
- DisCO can use COIN-OR's CGL when COLA is chosen as solver.

Valid Inequalities for MISOCO

We have the following cuts for MISOCO problems,

- Conic mixed-integer rounding (MIR) cuts given by [Atamturk and Narayanan(2010)] for general mixed integer case,
- Conic Gomory cuts given by [Çezik and Iyengar(2005)] for mixed 0–1 problems,
- Convex cuts defined by [Stubbs and Mehrotra(1999)] for mixed 0–1 convex problems,
- **D**isjunctive **c**onic **c**uts (DCC) and **d**isjunctive **cy**lindirical **c**uts (DCyC) defined by [Belotti et al.(2013)Belotti, Góez, Pólik, Ralphs, and Terlaky] for general mixed integer case,
- Two term disjunctions defined by [Kılınç-Karzan and Yıldız(2014)] for general mixed integer case.

Disjunctive Cuts by Belotti et al.

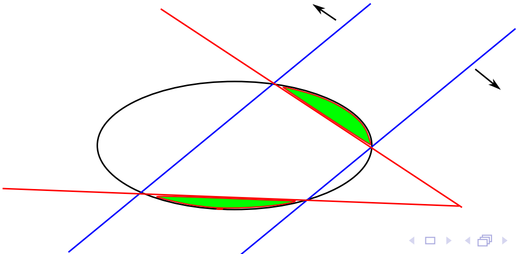
SOCO feasible region is given by

$$Ax = b$$
$$x \in \mathbb{L}^1 \times \cdots \times \mathbb{L}^k.$$

We convert the problem into the following form using $x = x^0 + Hw$,

$$w^\top Q^i w + 2q^{i\top} w + \rho^i \leq 0 \quad i \in \{1, \ldots, k\}$$
$$a^{i\top} x \geq \alpha^i \quad i \in \{1, \ldots, k\},$$

where H is the null space basis of A and x^0 is any point such that $Ax^0 = b$.



Conic MIR

Assume conic constraints in the following form,

$$\|Ax + Gy - b\| \leq d^\top x + e^\top y - h.$$

Introduce variables $(t_1, t_{2,m+1}) \in \mathbb{R} \times \mathbb{R}^m$,

$$\begin{aligned} t_1 &\leq d^\top x + e^\top y - h, \\ t_{i+1} &\geq |a_i x + g_i y - b_i| \quad i = 1, \ldots, m, \\ t_1 &\geq \|t_{2:m+1}\|. \end{aligned}$$

For a fixed row we define following set,

$$\mathcal{S} := \{(x, y, t) \in \mathbb{Z}_+^n \times \mathbb{R}_+^p \times \mathbb{R} : t \geq |ax + gy - b|\}.$$

Following is an MIR inequality for $\mathcal{S}$ for $\alpha > 0$,

$$\sum_{j=1}^{n} \varphi_{f_\alpha}\left(\frac{a_j}{\alpha}\right) x_j - \varphi_{f_\alpha}\left(\frac{b}{\alpha}\right) \leq \frac{(t + y^+ - y^-)}{|\alpha|}.$$

Conic MIR implementation

Cones in CBLIB are in the following form,

$$x_1 \geq \|x_{2:n}\|$$

We can lift this formulation as follows,

$$t \geq |x_j|$$
$$x_1 \geq \|(x_2, \ldots, t, \ldots, x_n)\|.$$

Assume one of the rows in constraint matrix implies $x_j = a^\top x + b^\top y + \beta$. Then generate MIR cut using following,

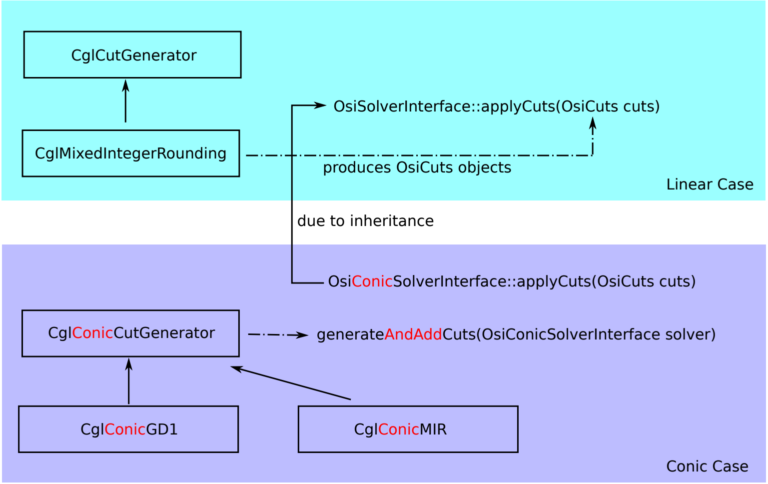
$$t \geq \left| a^\top x + b^\top y + \beta \right|.$$

Modify cone constraint as,

$$x_1 \geq \|(x_2, \ldots, t, \ldots, x_n)\|.$$

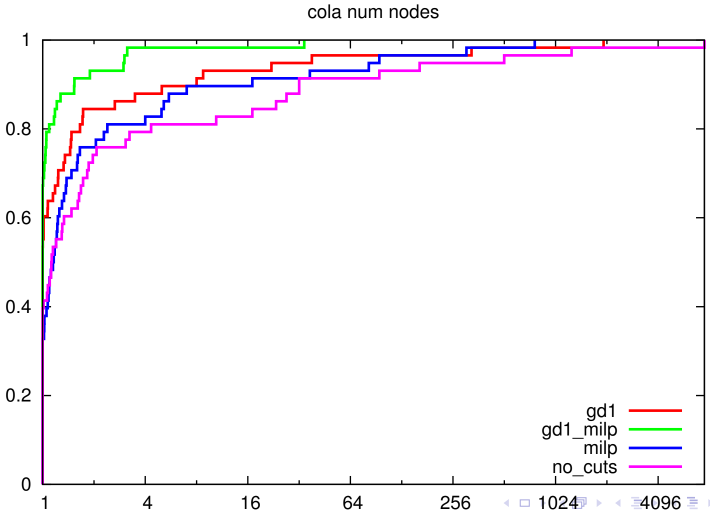
Implementing Conic Cuts

Figure: Conic CGL's relationship to COIN-OR projects



COLA with various cut strategies

Figure: COLA cut strategies, number of nodes



DisCO Branching Experiments with COLA–Nodes

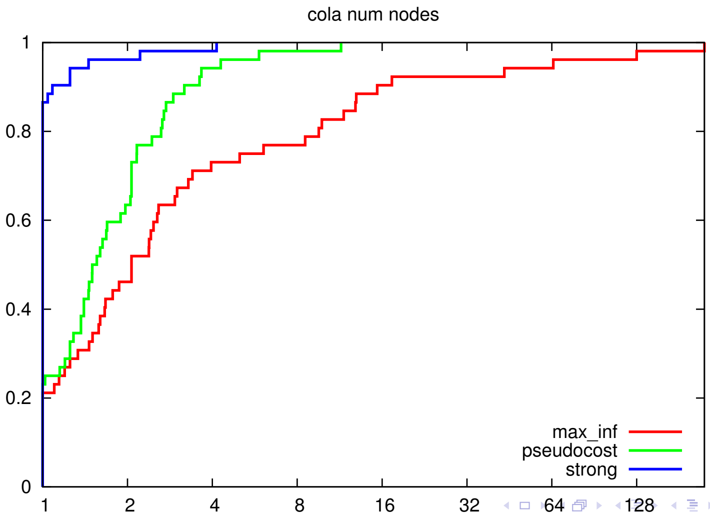
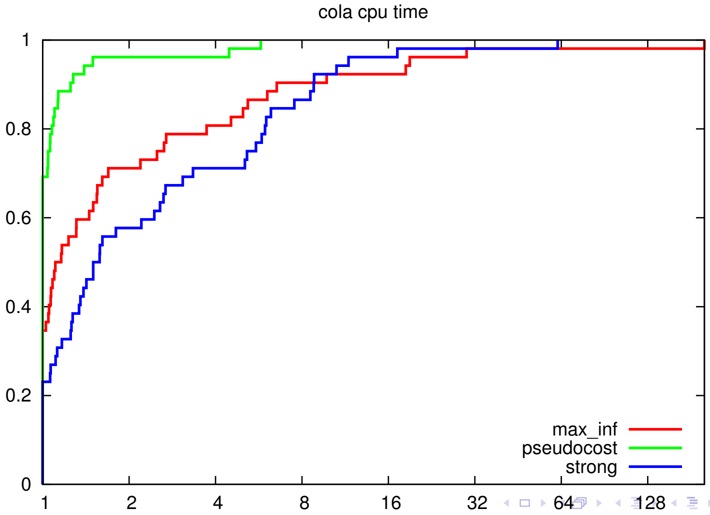


Figure: COLA branching strategy number of nodes

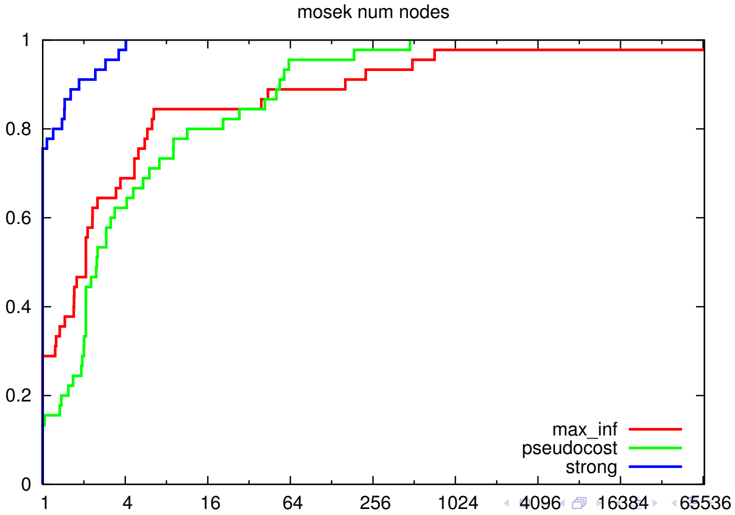
DisCO Branching Experiments with COLA–CPU time

Figure: COLA branching strategy CPU time



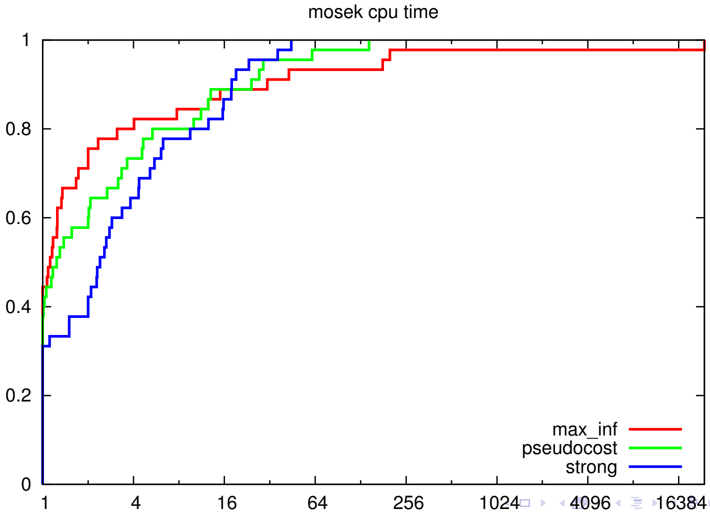
DisCO Branching Experiments with Mosek–Nodes

Figure: Mosek branching strategy, number of nodes



DisCO Branching Experiments with Mosek–CPU time

Figure: Mosek branching strategy, CPU time



Contributions

- Outer approximation algorithm performance results on continuous problems.
- Testing outer approximation algorithm on discrete problems in a branch and bound framework.
- Implementation details for conic MIR.
- Software tools conic OSI, interface for Mosek, COLA, DisCO.
- Comparing performance of outer approximation method to IPM in branch and bound framework.
- Comparison of different branching strategies for MISOCO.
- Performance of disjunctive cuts given by Belotti et. al.

Clone, Try, Contribute

`https://github.com/aykutbulut`
`https://github.com/coin-or`

References

Alper Atamturk and Vishnu Narayanan.
Conic mixed-integer rounding cuts.
Mathematical Programming, 122(1):1–20, 2010.
ISSN 0025-5610.
doi: 10.1007/s10107-008-0239-4.
URL http://dx.doi.org/10.1007/s10107-008-0239-4.

Pietro Belotti, Julio C. Góez, Imre Pólik, Ted K. Ralphs, and Tamás Terlaky.
On families of quadratic surfaces having fixed intersections with two hyperplanes.
Discrete Applied Mathematics, 161(16–17):2778–2793, 2013.
ISSN 0166-218X.

M.T. Çezik and G. Iyengar.
Cuts for mixed 0-1 conic programming.
Mathematical Programming, 104(1):179–202, 2005.
ISSN 0025-5610.
doi: 10.1007/s10107-005-0578-3.
URL http://dx.doi.org/10.1007/s10107-005-0578-3.

Henrik A. Friberg.
Cblib 2014: A benchmark library for conic mixed-integer and continuous optimization.
Optimization Online, 2014.
URL http://www.optimization-online.org/DB_FILE/2014/03/4301.pdf.

Julio C. Góez.
Mixed Integer Second Order Cone Optimization Disjunctive Conic Cuts: Theory and experiments.
PhD thesis, Lehigh University, Bethlehem, PA, September 2013.

Fatma Kılınç-Karzan and Sercan Yıldız.
Two-term disjunctions on the second-order cone.
In Jon Lee and Jens Vygen, editors, *Integer Programming and Combinatorial Optimization*, volume 8494 of *Lecture Notes in Computer Science*, pages 345–356. Springer International Publishing, 2014.

End of presentation

This is end of presentation!

Thank you for listening!

Maximum Infeasibility Branching–Nodes

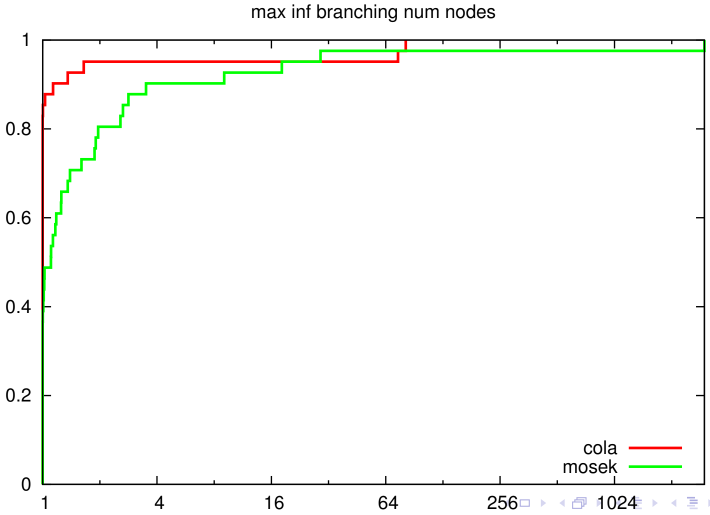


Figure: COLA vs Mosek with maximum infeasibility, number of nodes

Maximum Infeasibility Branching–CPU time

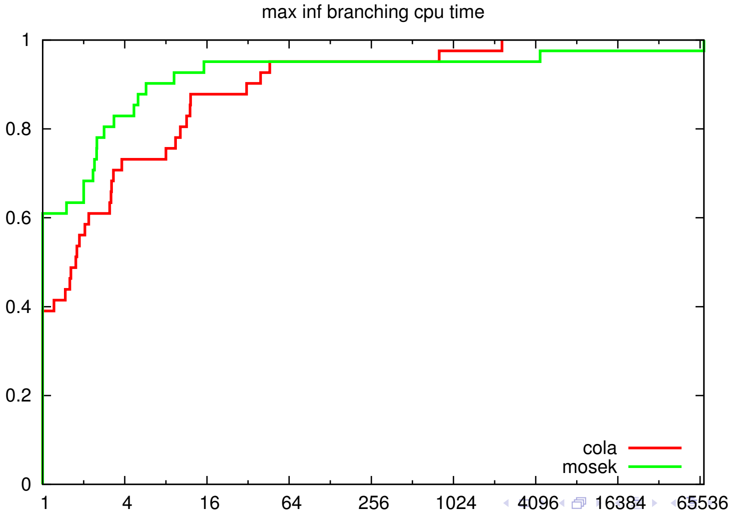
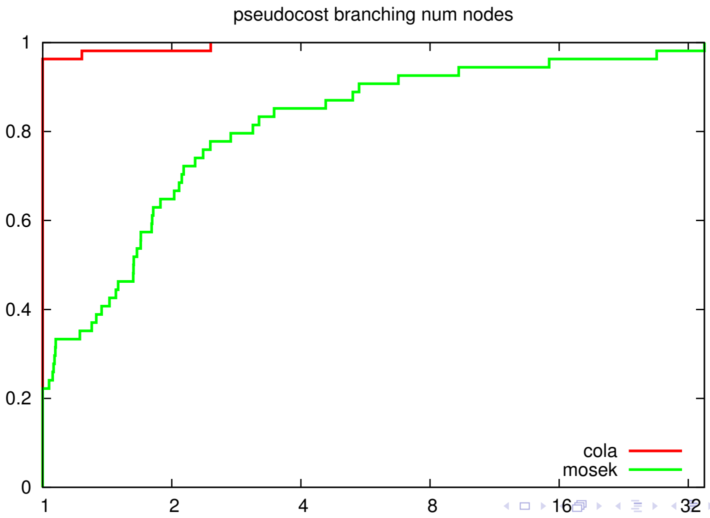


Figure: COLA vs Mosek with maximum infeasibility, CPU time

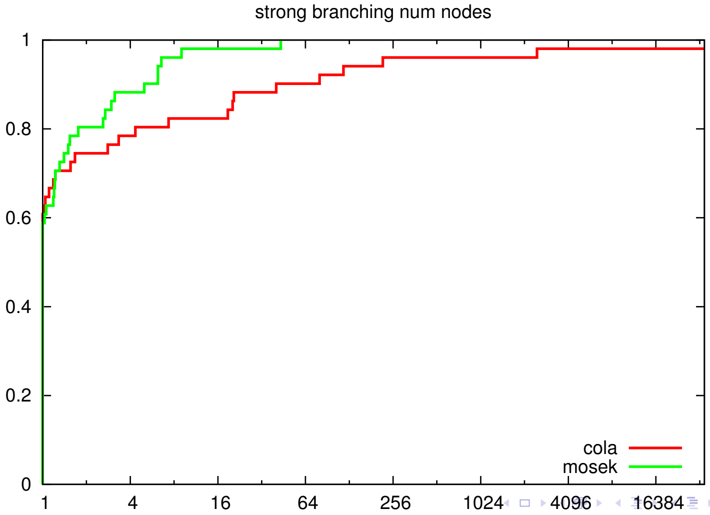
Pseudocost Branching

Figure: COLA vs Mosek with pseudocost, number of nodes



Strong Branching–Nodes

Figure: COLA vs Mosek with strong branching, number of nodes



Strong Branching–CPU time

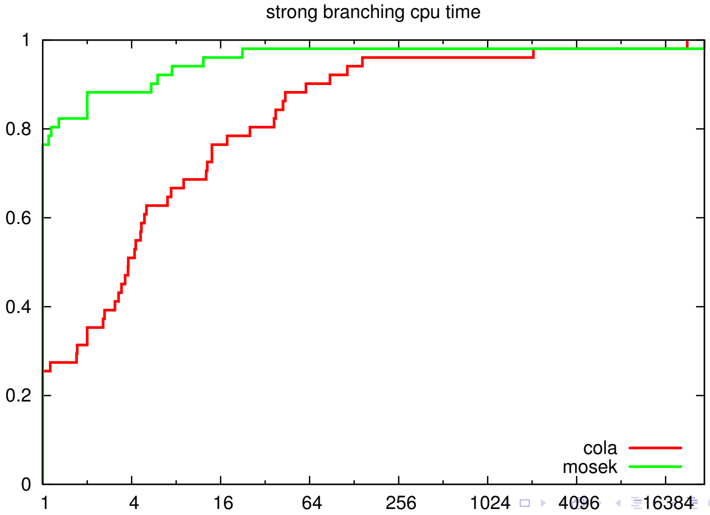


Figure: COLA vs Mosek with strong branching, CPU time

COLA–Pseudocost vs Mosek–Max Inf, Nodes

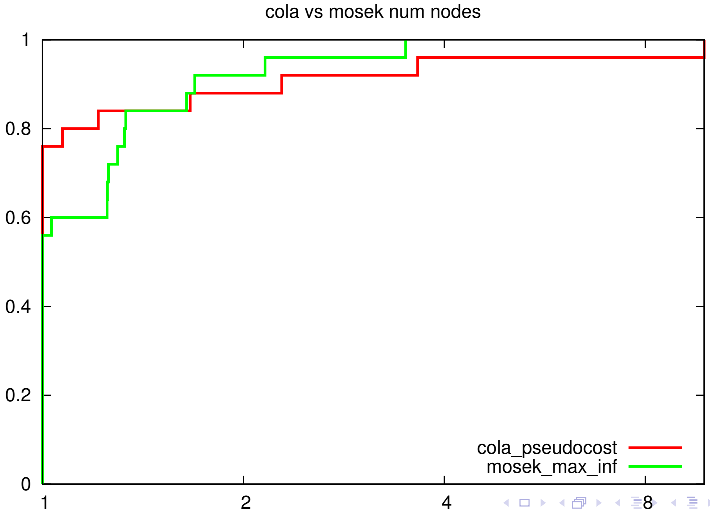


Figure: COLA pseudocost vs Mosek maximum infeasibility, number of nodes

COLA–Pseudocost vs Mosek–Max Inf, CPU time

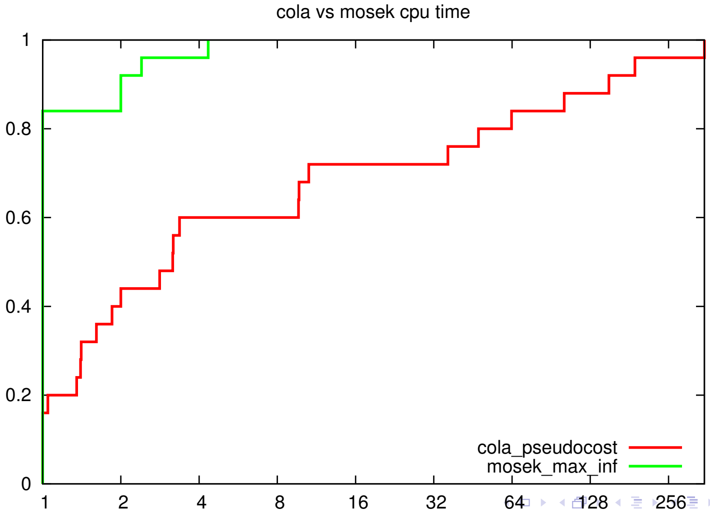


Figure: COLA pseudocost vs Mosek maximum infeasibility, CPU time