Computational advances in RLT algorithms: RAPOSa, a freely available implementation

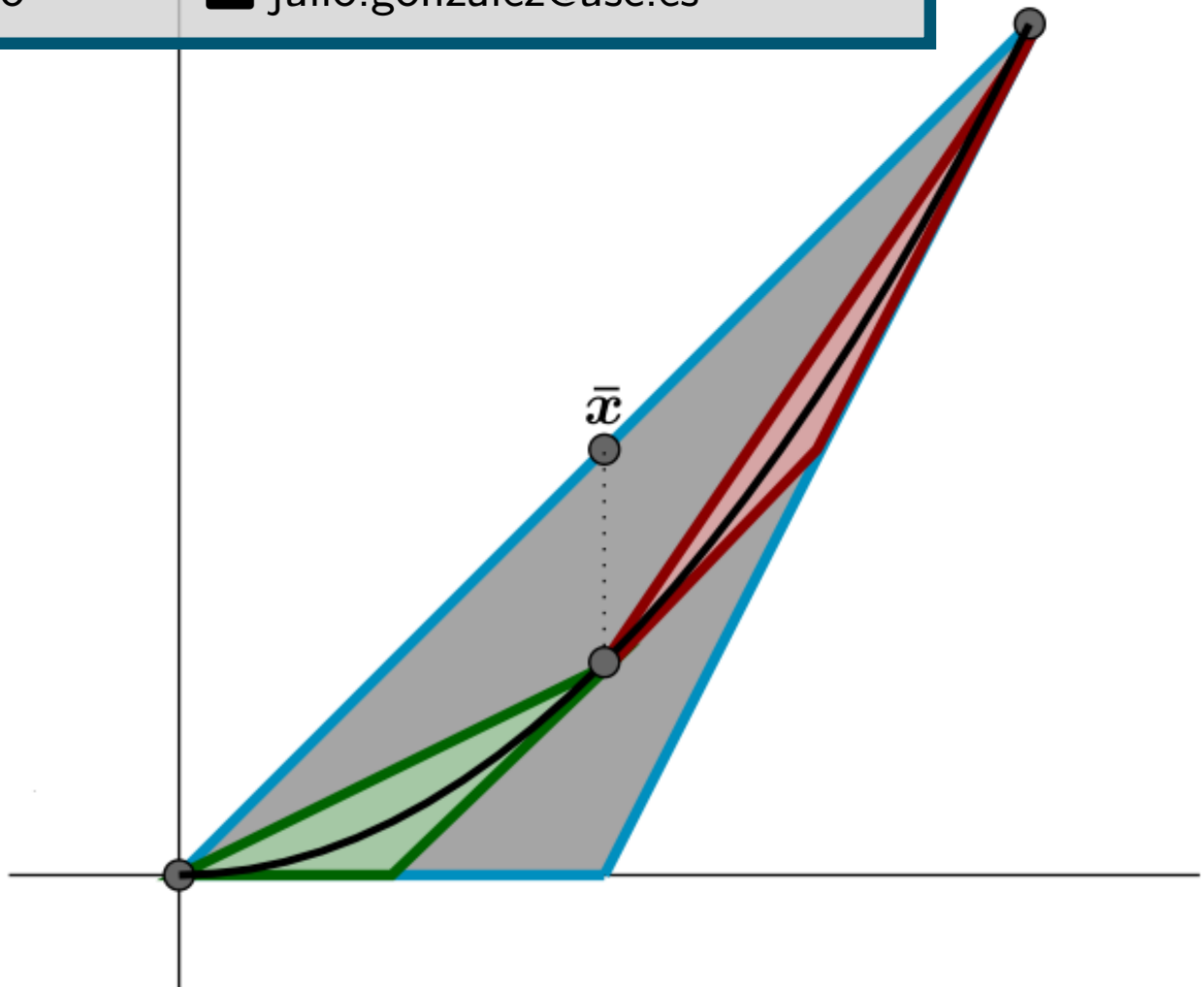
Julio González Díaz

Department of Statistics, Mathematical Analysis and Optimization. University of Santiago de Compostela
Modestya Research Group and IMAT. ITMATI affiliated researcher

eio.usc.es/pub/julio julio.gonzalez@usc.es

ISMP, Bordeaux

July 6th, 2018



Goal: Algorithm to (globally) solve polynomial optimization problems

Definition

A (continuous) polynomial optimization problem has the following form:

$$\begin{array}{rll} \text{minimize} & \phi_0(\boldsymbol{x}) & \\ \text{subject to} & \phi_r(\boldsymbol{x}) \geqslant \beta_r, & r = 1, \ldots, R_1 \\ & \phi_r(\boldsymbol{x}) = \beta_r, & r = R_1 + 1, \ldots, R \\ & \boldsymbol{x} \in \Omega \subset \mathbb{R}^n. & \end{array}$$

- $N = \{1, \ldots, n\}$ denotes the set of variables.
- Each $\phi_r(\boldsymbol{x})$ is a polynomial of degree δ_r.
- $\Omega = \{\boldsymbol{x} \in \mathbb{R}^n : 0 \leqslant l_j \leqslant x_j \leqslant u_j < \infty \ \forall j \in N\} \subset \mathbb{R}^n$ is a hyperrectangle containing the feasible region.
- Degree of the problem: $\delta = \max_{r \in \{1,\ldots,R\}} \delta_r$.

RLT basics

Reformulation-Linearization Technique for polynomial optimization
Sherali and Tuncbilek (1992)

Ingredients needed for RLT relaxations

- Hyperrectangle. $\Omega = \{\boldsymbol{x} \in \mathbb{R}^n : 0 \leqslant l_j \leqslant x_j \leqslant u_j < \infty \ \forall j \in N\} \subset \mathbb{R}^n$
- Bound-factors, the basis for McCormick type of inequalities $(x_j - l_j) \geqslant 0$ and $(u_j - x_j) \geqslant 0, \forall j \in N$
- Bound-factor constraints. Given $N^\delta = N \times \overset{\delta}{\ldots} \times N$,
  $F_\delta(J_1, J_2) = \prod_{j \in J_1} (x_j - l_j) \prod_{j \in J_2} (u_j - x_j) \geqslant 0, \forall J_1 \times J_2 \subset N^\delta$
- RLT-variables, defined through the RLT-defining identities
  $X_J = \prod_{j \in J} x_j, \ \forall J \subset N^\delta, 2 \leqslant |J| \leqslant \delta$

Linearization

LP subproblems at the core of the algorithm

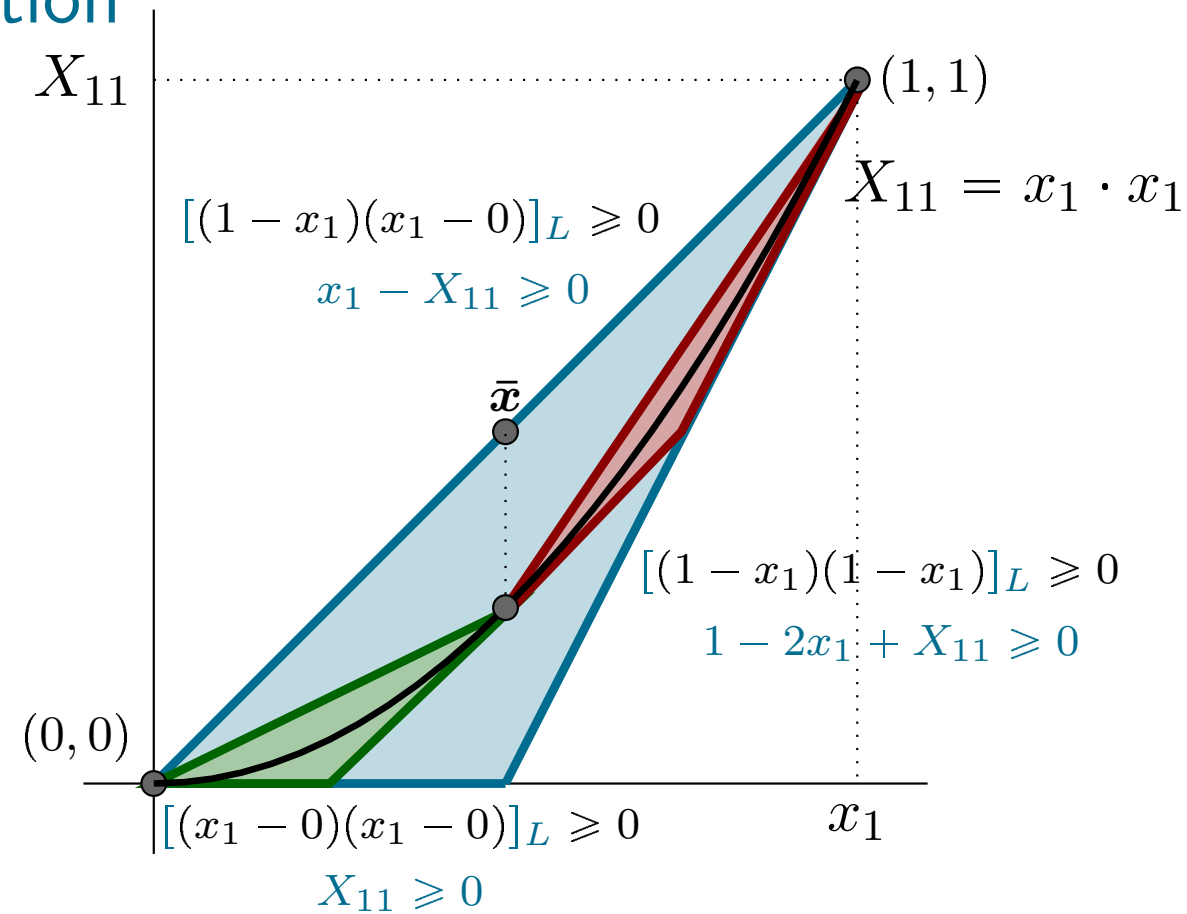
- Add the bound factor constraints to the original formulation
- Replace all the monomials with the corresponding RLT-variables
  (this relaxation is denoted by $[\cdot]_L$)

RLT linearization

$$
\begin{aligned}
\text{minimize} \quad & [\phi_0(\boldsymbol{x})]_L \\
\text{subject to} \quad & [\phi_r(\boldsymbol{x})]_L \geqslant \beta_r, && r = 1, \ldots, R_1 \\
& [\phi_r(\boldsymbol{x})]_L = \beta_r, && r = R_1 + 1, \ldots, R \\
& [F_\delta(J_1, J_2)]_L \geqslant 0, && \forall J_1 \cup J_2 \subset N^\delta, |J_1 \cup J_2| = \delta \\
& X_J = \textstyle\prod_{j \in J} x_j && \forall J \subset N^\delta \\
& \boldsymbol{x} \in \Omega \subset \mathbb{R}^n.
\end{aligned}
$$

- Bound factors are added
- Polynomials are linearized
- RLT-defining identities make this problem equivalent to the original one. They are not part of the relaxation
- Branch and bound algorithm:
  - Branching on variables appearing in violated RLT-defining identities ("McCormick cuts")

RLT intuition



- Bound-factor constraints $\Rightarrow$ approximation of underlying polynomials
- Solution of relaxed problem $\Rightarrow$ Lower Bounds
- Violation of RLT-defining identities $\Rightarrow$ branching (tighter relaxations)
- Convergence to global optimum guaranteed

RLT enhancements

Main enhancements from the literature

- Use of local NLP solver.
  - Allows to obtain Upper Bounds for the B&B process
- SDP-cuts. Sherali, Dalkiran, and Desai (2012)
  - Adding linear constraints based on violations on positive semidefinitiness constraints
- Reduced-RLT. Sherali, Dalkiran, and Liberti (2012)
  - Reducing the size of the relaxations in the presence of linear equality constraints
- J-sets. Dalkiran and Sherali (2013)
  - Reducing the size of the relaxations in relatively sparse problems (many monomials of the degree of the problem are not present)

A new implementation: RAPOSa

Project team

Reformulation Algorithm for Polynomial Optimization Santiago

Project started around 2 years and a half ago

Developing team formed by researchers from

- University of Santiago de Compostela
  - Julio González-Díaz
  - Brais González-Rodríguez
  - Ángel M. González-Rueda
  - David Rodríguez-Penas
- ITMATI (Technological Institute for Industrial Mathematics)
  - Joaquín Ossorio-Castillo
  - Diego Rodríguez-Martínez

Launching first downloadable version (free software)

A new implementation: RAPOSa

Implementation background

- Core of the implementation in C++
- Windows version available for download
  (Linux coming soon)
- Compatible with standard .nl files
  (they can be created with AMPL, Pyomo)
- Working on direct availability via AMPL and via NEOS server
- Still in beta version (feedback from users needed)

A new implementation: RAPOSa

Current implementation (RAPOSa 0.7.0)

Features from the literature

- Implementation of basic RLT algorithm
  (Sherali and Tuncbilek, 1992)
- Free and commercial LP solvers
  (currently lpSolve and Gurobi)
- Interaction with free and commercial NLP solvers (via .nl files)
  (currently Ipopt and Knitro)
- J-sets
  (Dalkiran and Sherali, 2013)
  Clearly superior to implementing all bound factor constraints

Novel features

- Warm generation of J-sets
- Warm start on LP solver (implemented only for Gurobi)
- Parallelization of the B&B algorithm

A new implementation: RAPOSa

Current implementation (RAPOSa 0.7.0)

Test sets

- Current test sets taken from Sherali, Dalkiran, and Liberti (2012). 180 problems of varying degree:
  - Problems of degrees 2 to 7 (30 problems each)
  - Randomly generated with varying densities
- Additional tests sets in preparation

  (in particular, taking problems from CUTEst)
- Additional test problems are welcome

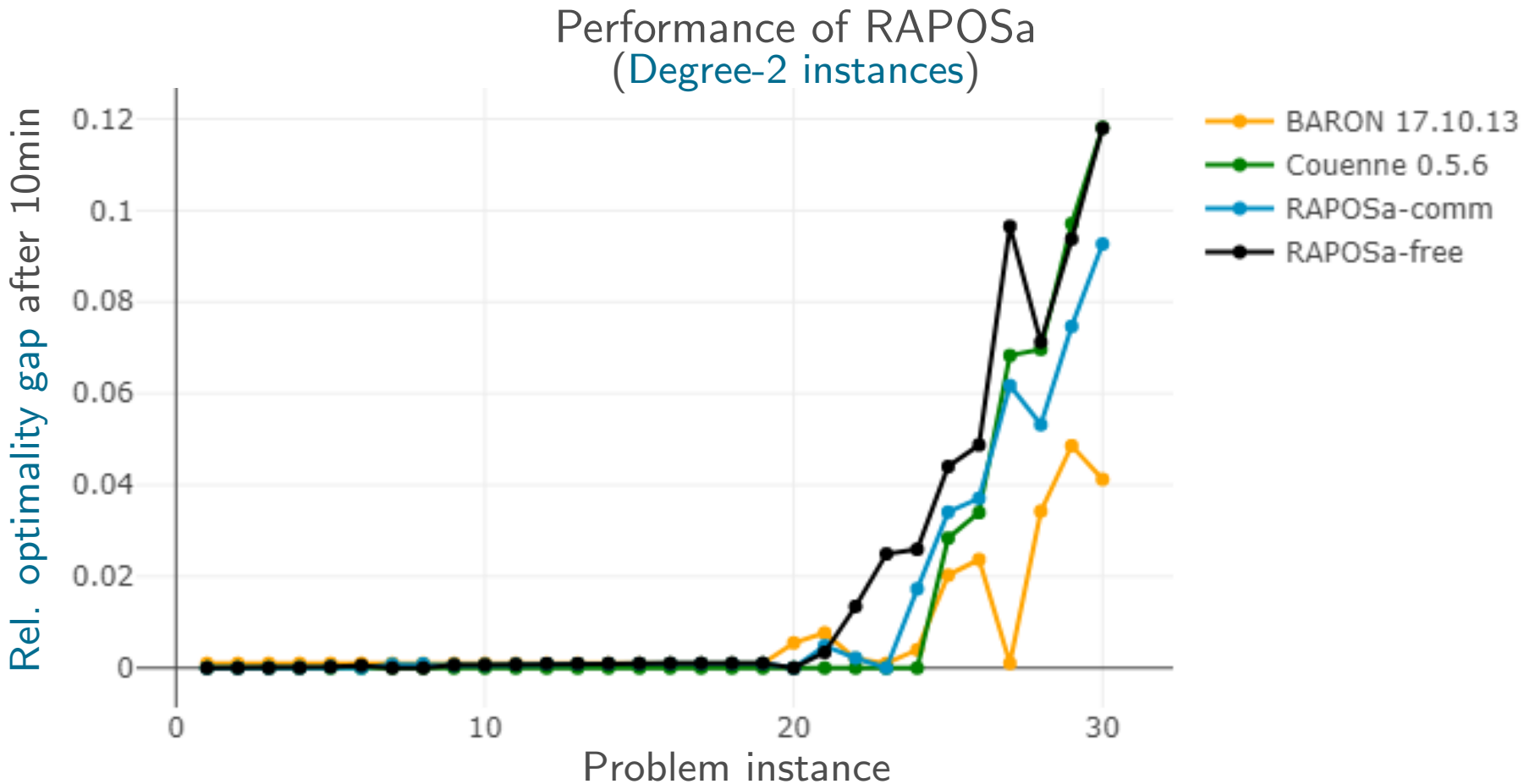
Tests were run on a high performance computing cluster: CESGA (Galician Supercomputing Center)

Preliminary results!!

A new implementation: RAPOSa

Basic implementation (without novel features)

Degree-2 instances



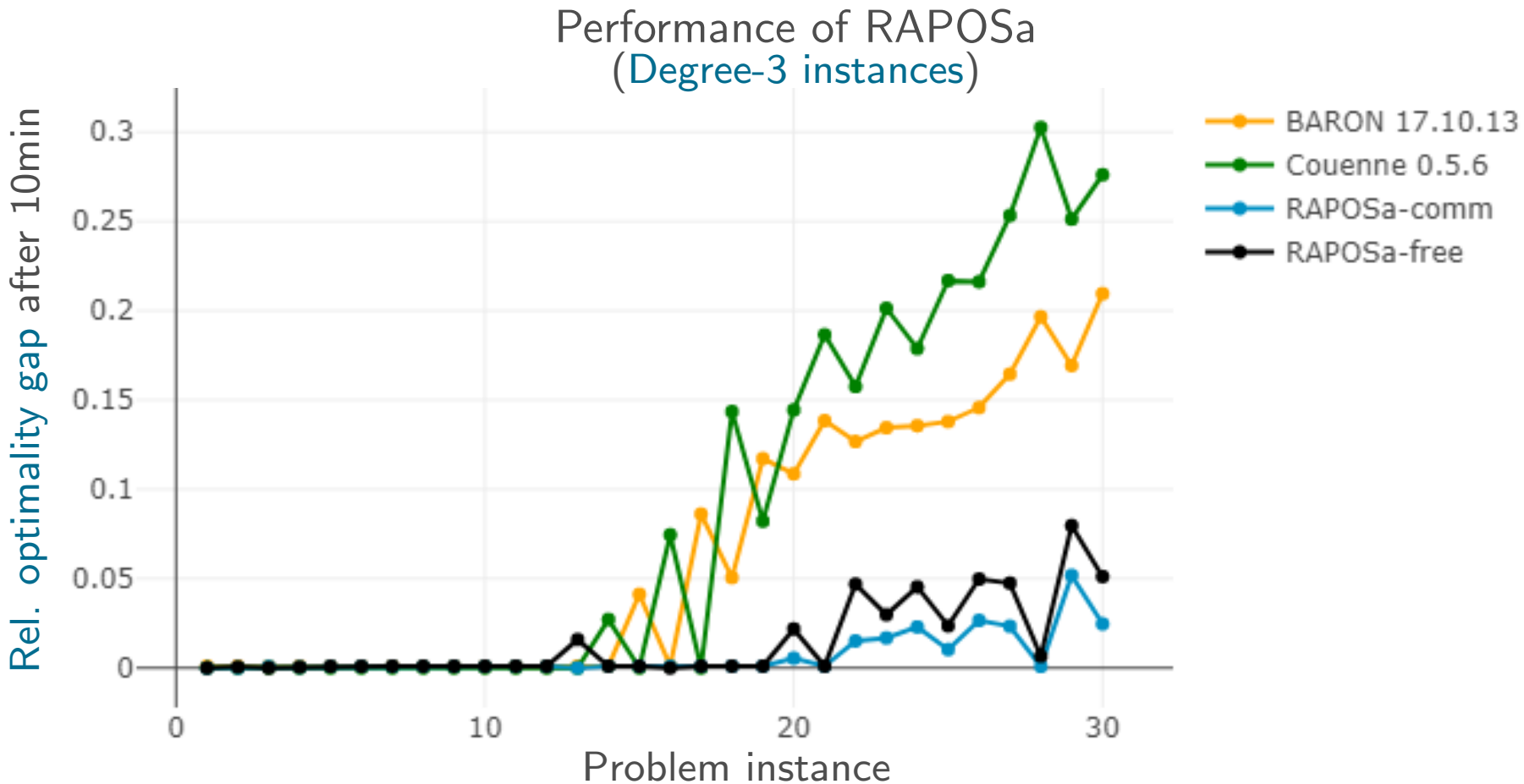
- RAPOSa-comm = Gurobi + Knitro
- BARON performs slightly better
- Not very big differences

RAPOSa-free = IpSolve + Ipopt

A new implementation: RAPOSa

Basic implementation (without novel features)

Degree-3 instances

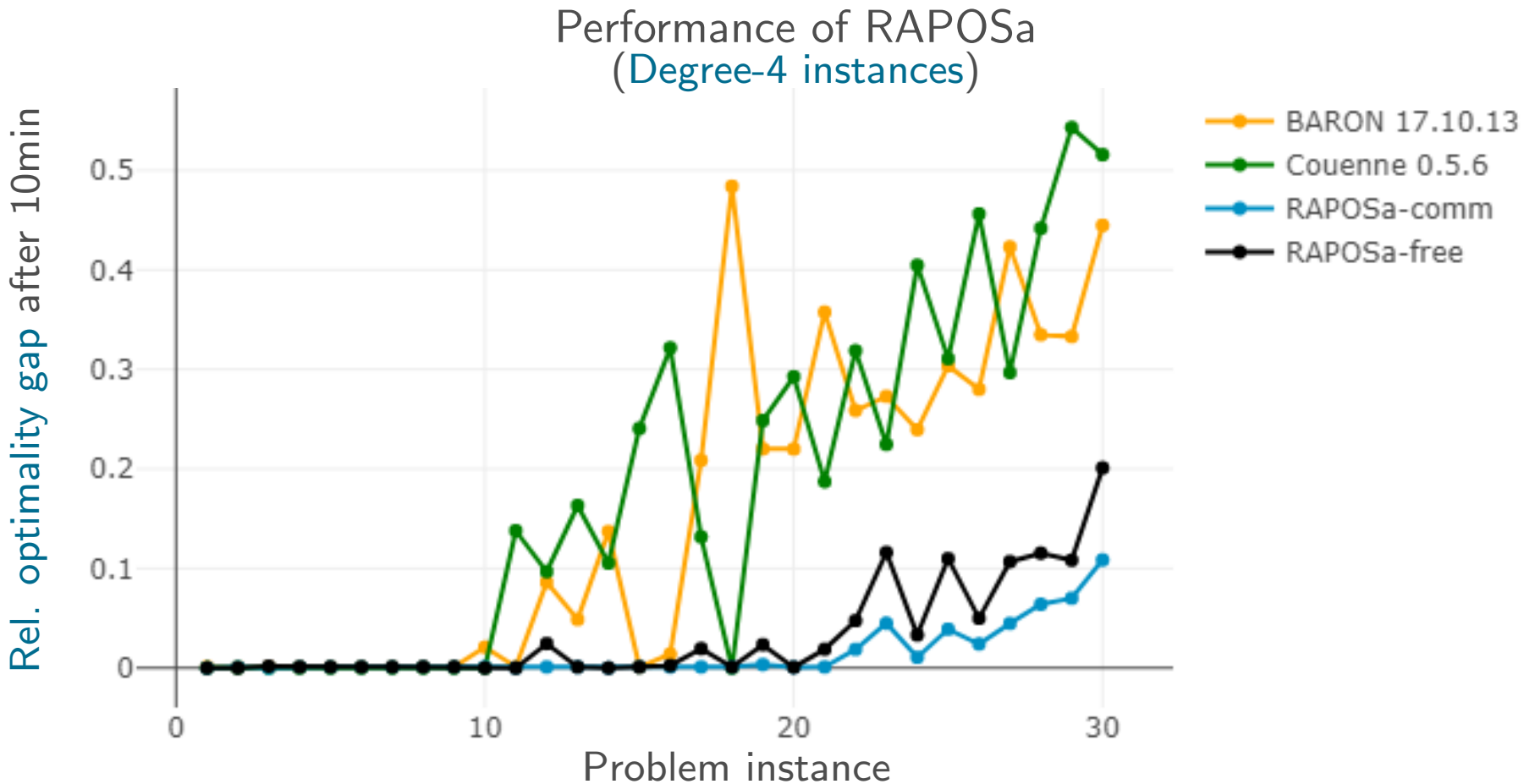


- RAPOSa-comm → Gurobi + Knitro
- RAPOSa-free → IpSolve + Ipopt
- Both versions of RAPOSa are preferable now
- More significant differences

A new implementation: RAPOSa

Basic implementation (without novel features)

Degree-4 instances

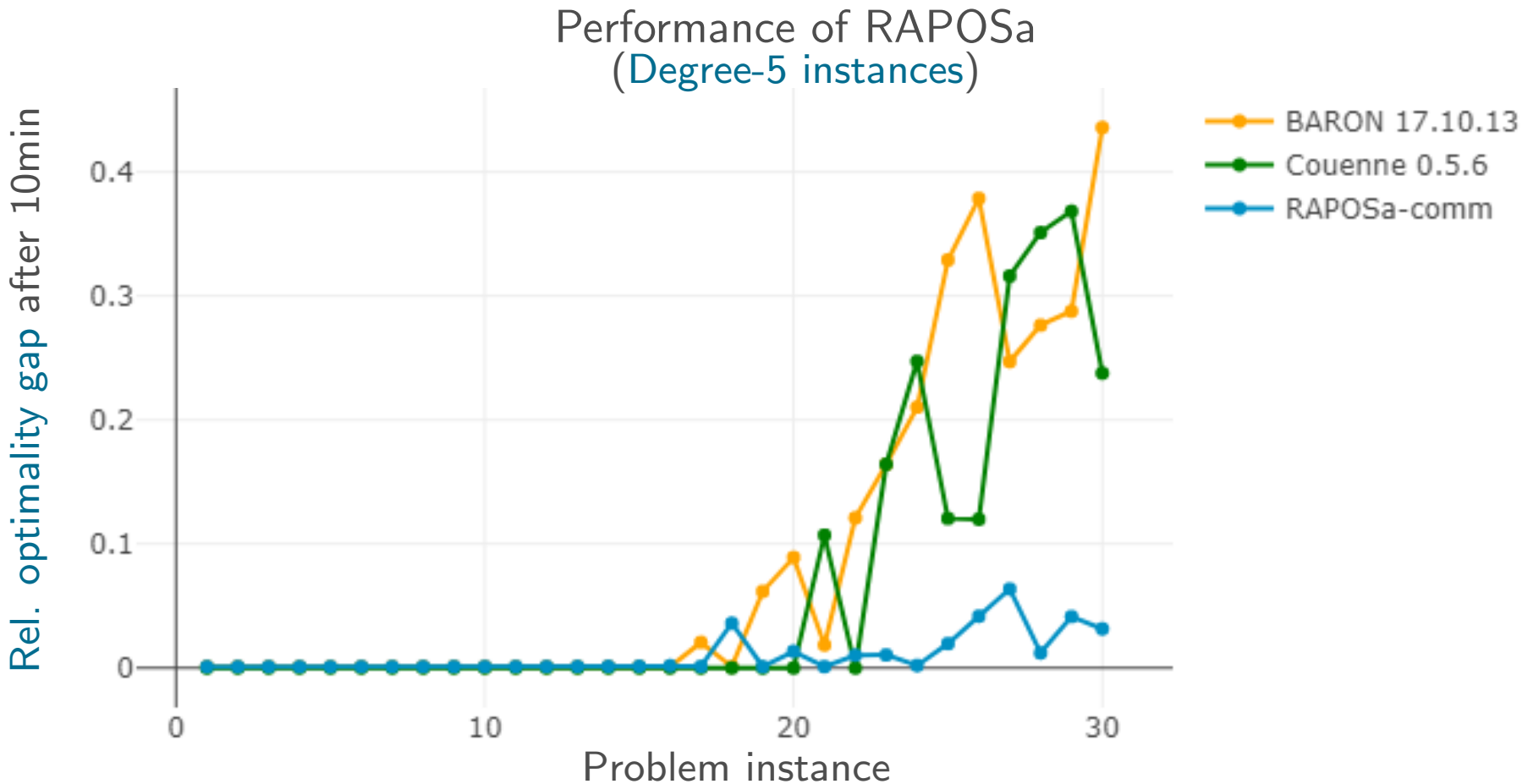


- RAPOSa-comm → Gurobi + Knitro
- RAPOSa-free → IpSolve + Ipopt
- Both versions of RAPOSa are preferable now
- More significant differences

A new implementation: RAPOSa

Basic implementation (without novel features)

Degree-5 instances

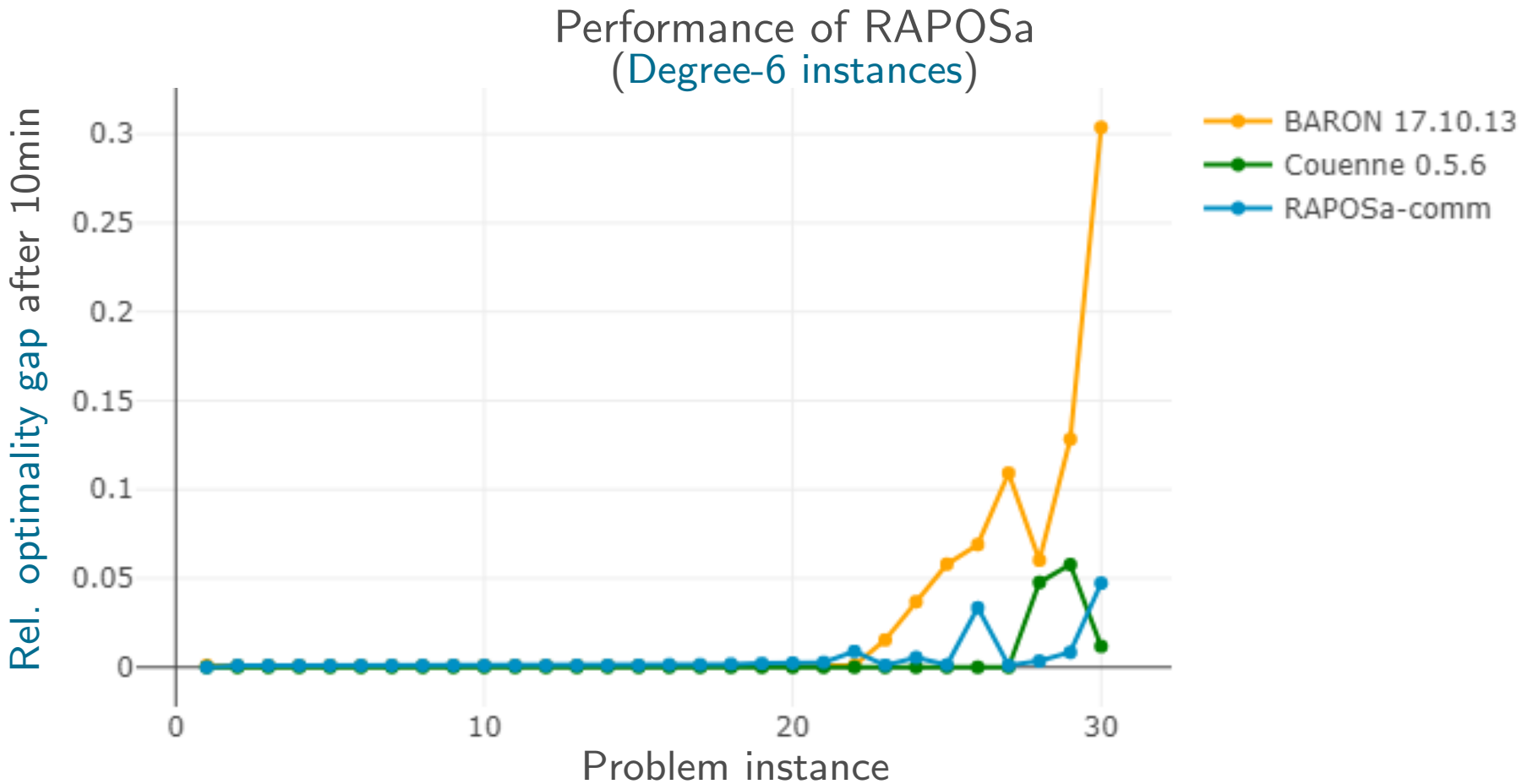


- RAPOSa-comm is still preferable
- Still significant differences
- IpSolve starts having problems to solve subproblems in some instances

A new implementation: RAPOSa

Basic implementation (without novel features)

Degree-6 instances

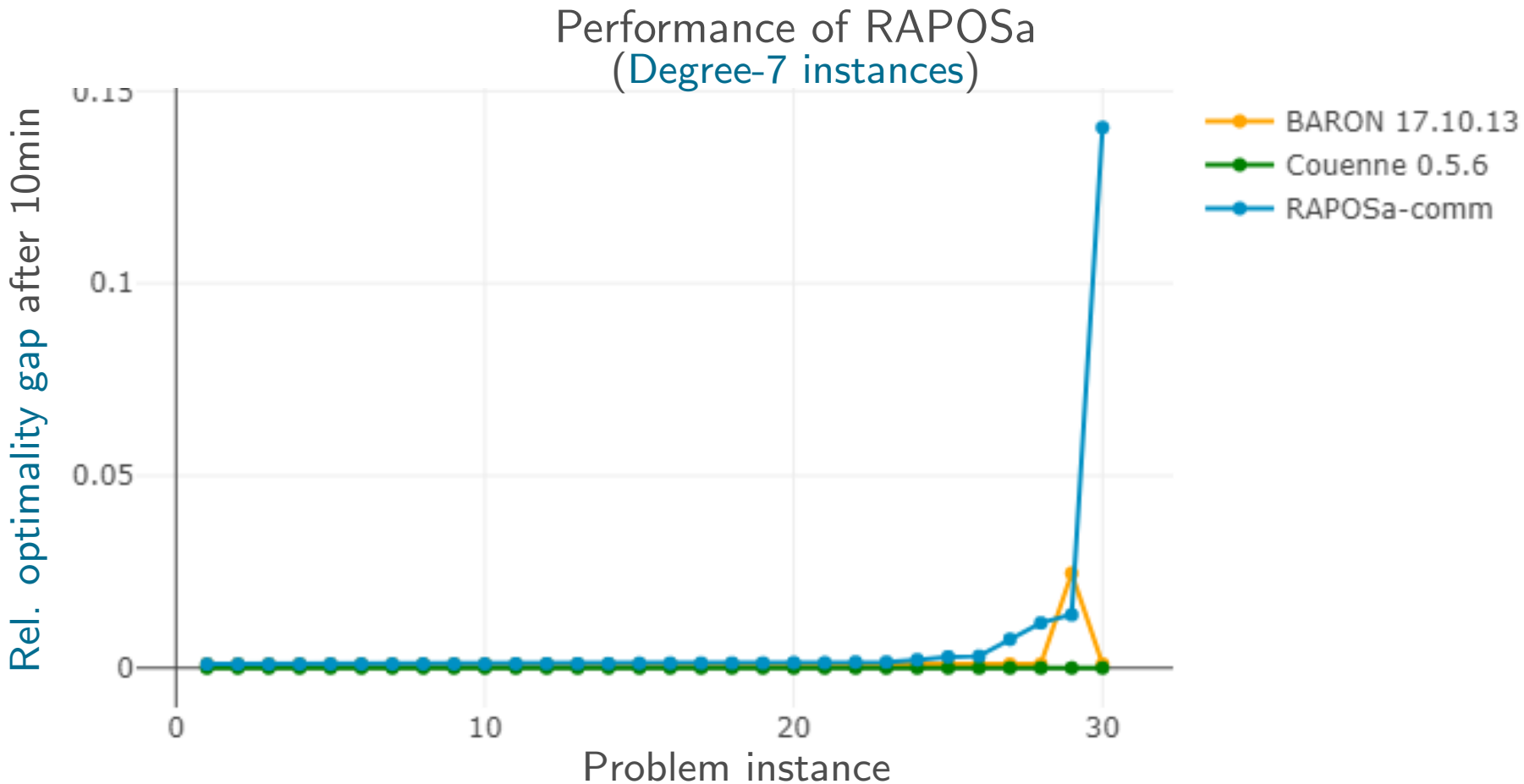


- Couenne seems to be catching up
- Why? We do not know

A new implementation: RAPOSa

Basic implementation (without novel features)

Degree-7 instances



- Couenne does better now. RAPOSa's performance is inferior (specially for one instance)
- Why? We do not know

A new implementation: RAPOSa

Current implementation (RAPOSa 0.7.0)

Promising starting point for a first release

Pure B&B algorithm

- No bound tightening
- No cuts

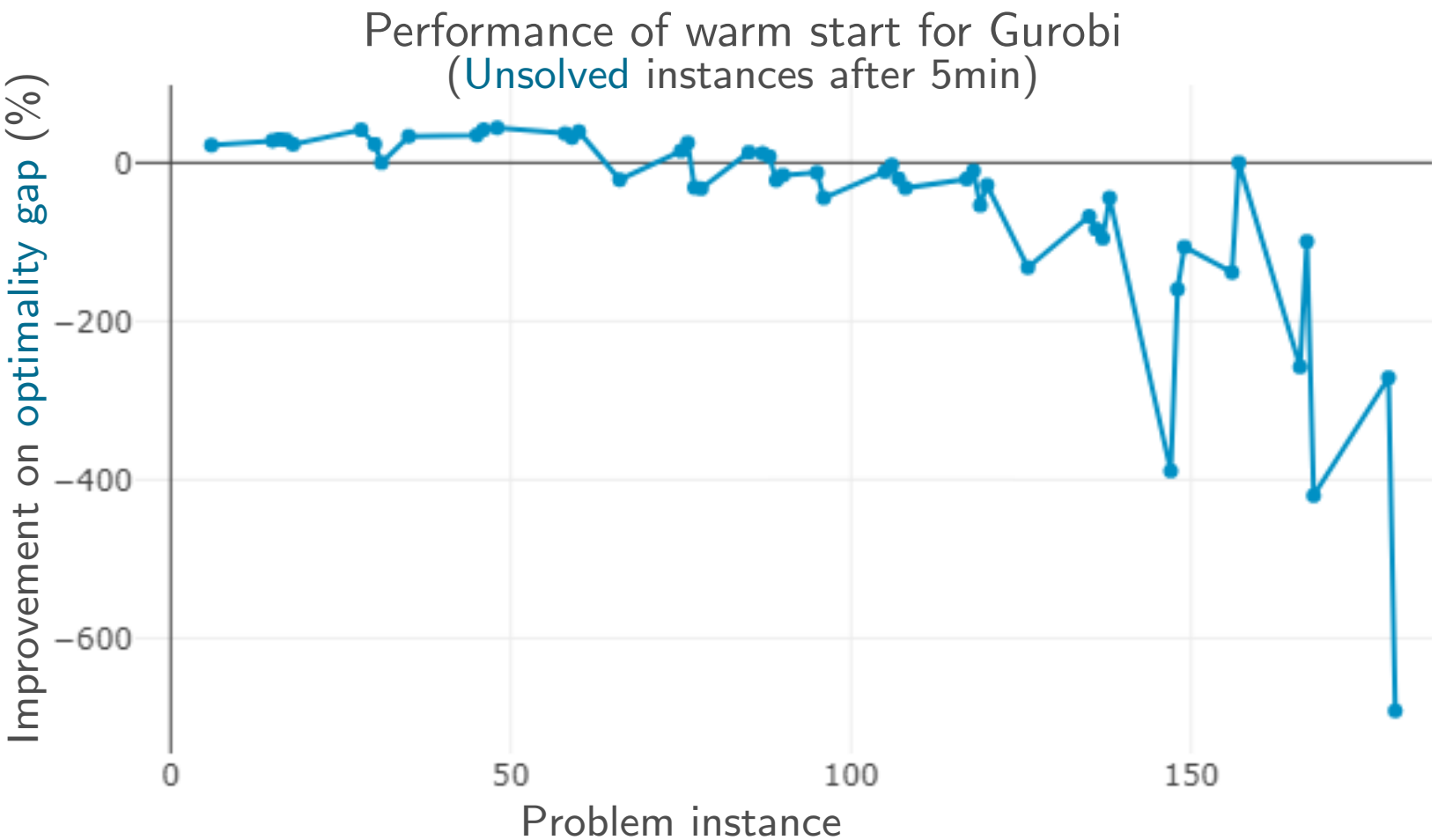
Novel features

- Warm start on LP solver (Gurobi)
- Warm generation of J-sets
- Parallelization

A new implementation: RAPOSa

Current implementation (RAPOSa 0.7.0)

Novel features	
Warm start LP	X
Warm J-set	
Parallelization	

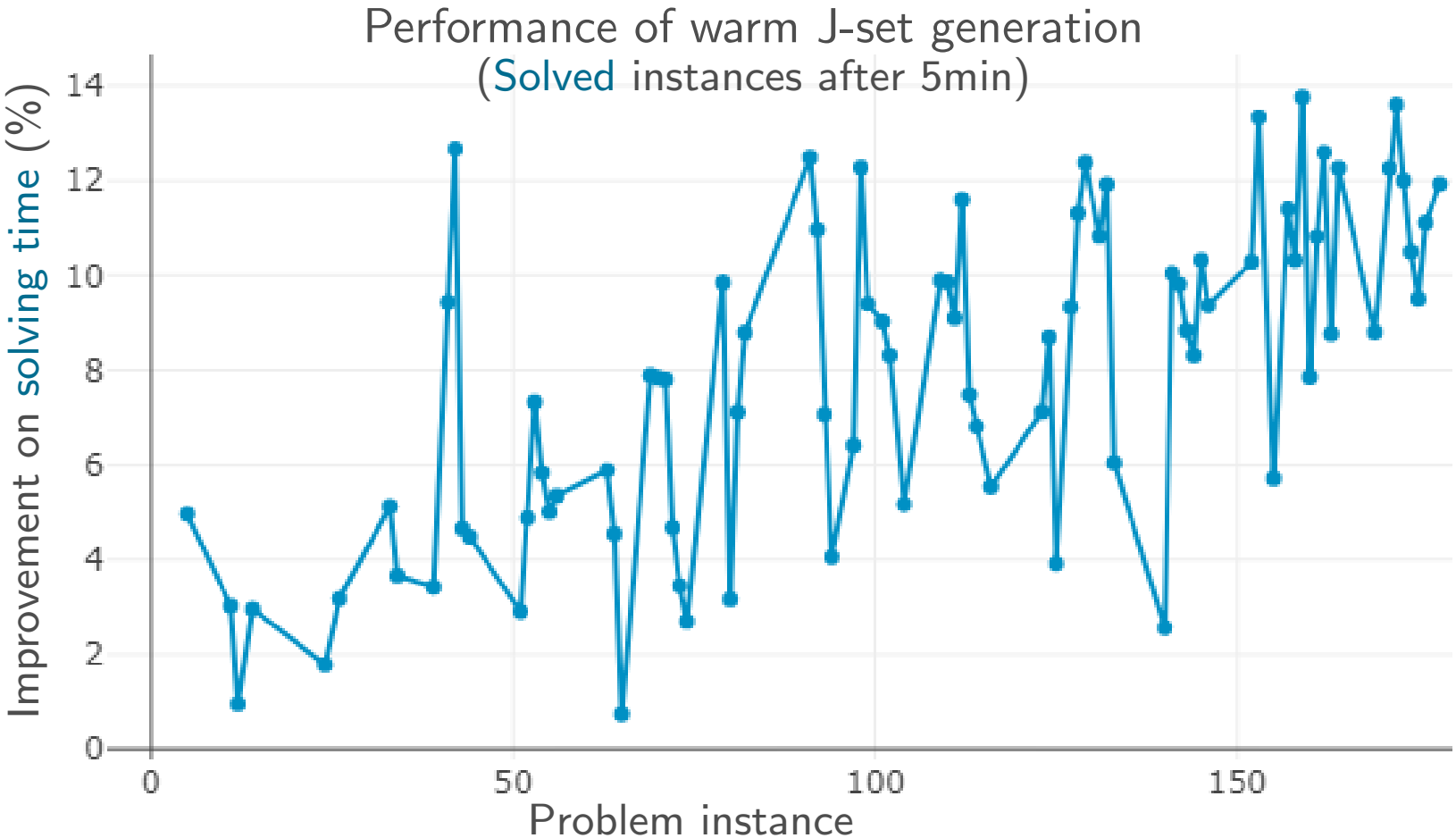


- Relative performance of RAPOSa 0.7.0 Vs. Basic implementation
- Basic implementation seems preferable (set as default configuration)
- Further analysis needed

A new implementation: RAPOSa

Current implementation (RAPOSa 0.7.0)

Novel features	
Warm start LP	✗
Warm J-set	✓
Parallelization	

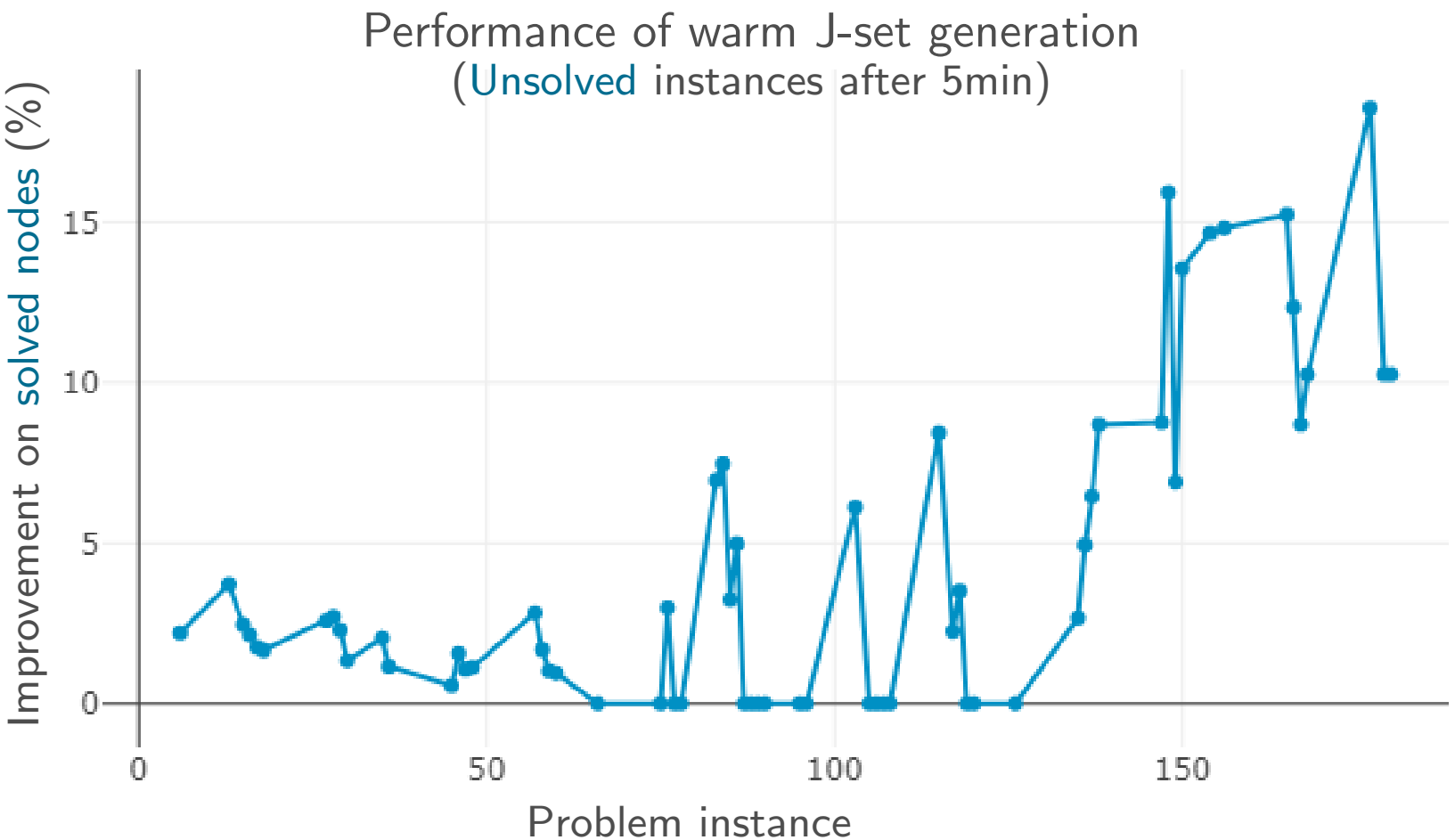


- Relative performance of RAPOSa 0.7.0 Vs. Basic implementation
- RAPOSa 0.7.0 better in all instances. Average improvement: **7.85%**

A new implementation: RAPOSa

Current implementation (RAPOSa 0.7.0)

Novel features	
Warm start LP	✗
Warm J-set	✓
Parallelization	



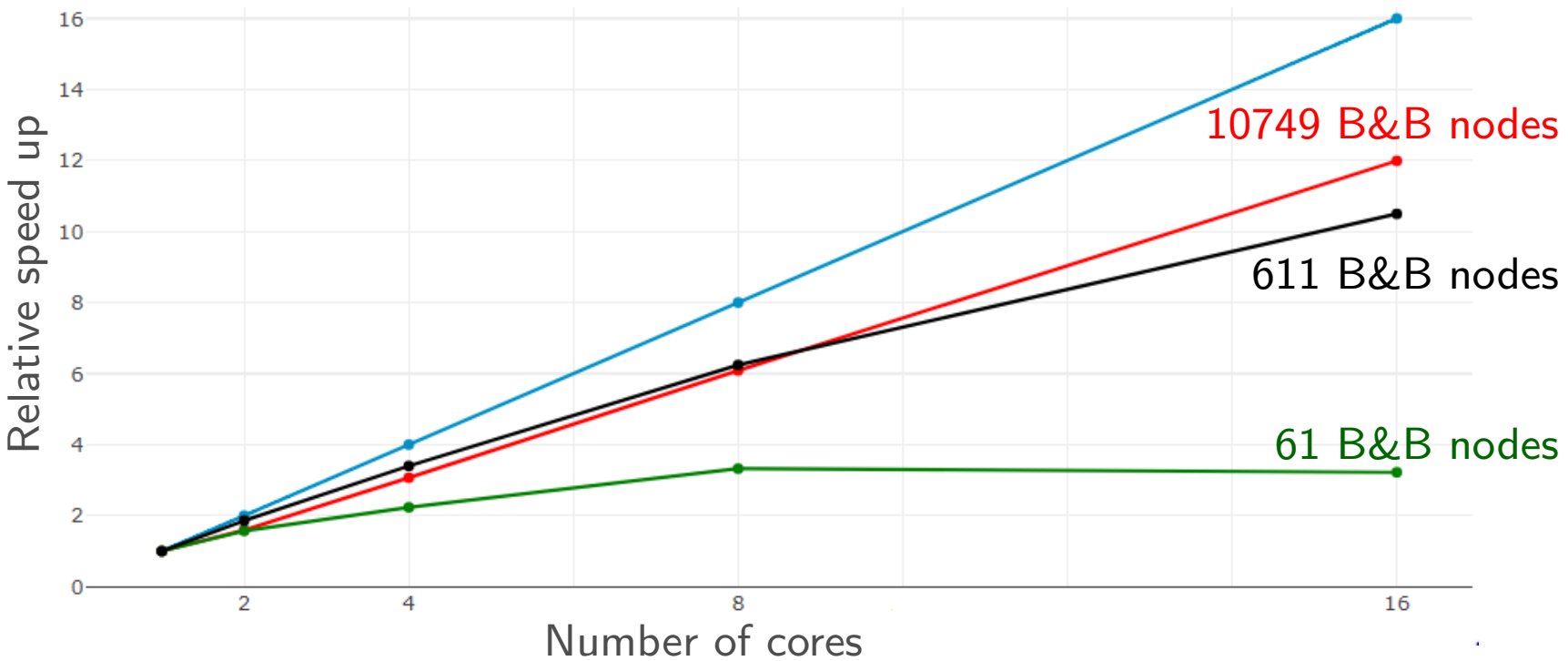
- Relative performance of RAPOSa 0.7.0 Vs. Basic implementation
- RAPOSa 0.7.0 better in all instances. Average improvement: $\mathbf{4.22\%}$

A new implementation: RAPOSa

Current implementation (RAPOSa 0.7.0)

Novel features	
Warm start LP	X
Warm J-set	✓
Parallelization	✓

Performance of parallel implementation
(on three representative instances)



- Performance of parallel RAPOSa 0.7.0 Vs. Basic implementation
- Promising scalability
- Contrast with BARON (improvements mainly for 2-cores and integer variables)

A new implementation: RAPOSa
Forthcoming enhancements and future approaches

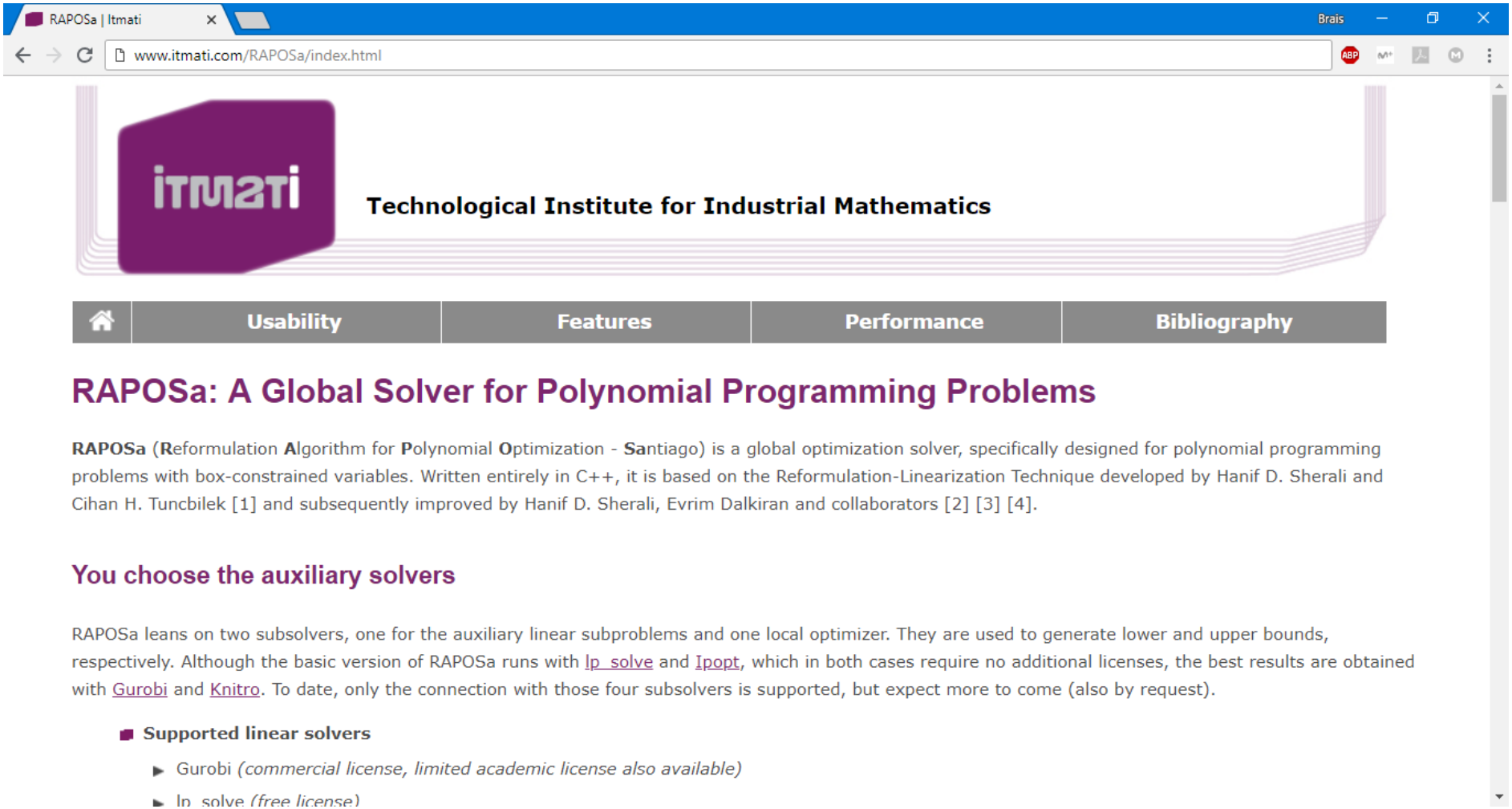
Forthcoming enhancements borrowed from the literature

- Constraint propagation. To improve bounds on variables
- SDP-cuts. Cuts based on semidefinite programming
- Reduced-RLT. Size reductions based on LP and LP duality

Future approaches based on new research

- SOCP-cuts. Study the potential of adding cuts based on second order cone programming
- Learning. Study the potential of learning techniques
  - To calibrate de algorithm based on the structure of the problem
  - Recalibrate along the B&B tree

Software availability



http://www.itmati.com/RAPOSa/index.html

julio.gonzalez@usc.es

Computational advances in RLT algorithms: RAPOSa, a freely available implementation

Julio González Díaz

Department of Statistics, Mathematical Analysis and Optimization. University of Santiago de Compostela
Modestya Research Group and IMAT. ITMATI affiliated researcher

eio.usc.es/pub/julio julio.gonzalez@usc.es

ISMP, Bordeaux

July 6th, 2018