Noname manuscript No.
(will be inserted by the editor)

Alternative Regularizations for Outer-Approximation Algorithms for Convex MINLP

David E. Bernal · Zedong Peng · Jan Kronqvist · Ignacio E. Grossmann

Received: date / Accepted: date

Abstract In this work, we extend the regularization framework from Kronqvist et al [36] by incorporating several new regularization functions and develop a regularized single-tree search method for solving convex mixed-integer nonlinear programming (MINLP) problems. We propose a set of regularization functions based on distance metrics and Lagrangean approximations, used in the projection problem for finding new integer combinations to be used within the Outer-Approximation (OA) method. The new approach, called Regularized Outer-Approximation (ROA), has been implemented as part of the open-source **M**ixed-**i**nteger **n**onlinear **d**ecomposition **t**oolbox for **Py**omo - MindtPy. We compare the OA method with seven regularization function alternatives for ROA. Moreover, we extend the LP/NLP Branch & Bound method proposed by Quesada and Grossmann [46] to include regularization in an algorithm denoted RLP/NLP. We provide convergence guarantees for both ROA and RLP/NLP. Finally, we perform an extensive computational experiment considering all convex MINLP problems in the benchmark library MINLPLib. The

David E. Bernal
Research Institute for Advanced Computer Science, Universities Space Research Association, Mountain View, CA, USA
Quantum Artificial Intelligence Laboratory (QuAIL), NASA Ames Research Center, Mountain View, CA, USA

Zedong Peng,
College of Control Science and Engineering, Zhejiang University, Hangzhou, China
Business Growth BU, JD.com, Beijing, China

Jan Kronqvist,
Optimization and Systems Theory, Department of Mathematics, KTH Royal Institute of Technology, Stockholm, Sweden
Department of Computing, Imperial College London, London, United Kingdom

David E. Bernal · Ignacio E. Grossmann
Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh PA, USA
E-mail: grossmann@cmu.edu

computational results show clear advantages of using regularization combined with the OA method.

Keywords Convex MINLP · Outer Approximation · Regularization for Mixed-Integer Optimization

1 Introduction

Optimization problems whose objective and constraints can be represented by algebraic linear and nonlinear functions of both continuous and discrete variables are commonly referred to as mixed-integer nonlinear programs (MINLP). MINLP is a highly versatile modeling paradigm, allowing even Universal Turing Machines to be encoded via a Minsky's register machine [39]. There is a large variety of practical applications and optimization tasks that can be modeled using MINLP, see *e.g.,* [8, 19, 37, 51]. Although MINLPs are non-convex optimization problems because of some variables' discreteness, the term convex MINLP is used to denote problems where the continuously relaxed problem is convex [34].

Convex MINLP problems are an important class of problems, as the convex properties can be exploited to derive efficient decomposition algorithms. Among these decomposition algorithms for MINLP, we have Branch & Bound (B&B) [13], Generalized Benders Decomposition [20], Outer-Approximation (OA) [16], Partial Surrogate Cuts [46], Extended Cutting Plane (ECP) [53], Feasibility Pump [4, 7] Extended Supported Hyperplanes (ESH) [32], and the center-cut [35] method. Moreover, the are several extensions of the OA method, such as the single-tree OA [46], Quadratic-cuts OA [49], conic-based OA [11], Decomposition-based OA [43], and Proximal OA [14]. These methods exploit the convex properties to derive valid linearizations of the nonlinear constraints based on their gradients. These linearizations are equivalent to first-order Taylor expansions of the nonlinear functions in the inequality constraints. They define a linear region that overestimates the problem's nonlinear feasible region because of the convexity property. Since methods such as ECP, ESH, and OA, in their original form, all solve MILP relaxation problems via a B&B search tree in each iteration, these methods are known as multi-tree methods [11, 42]. Algorithm developers have actively interested in solving MINLP problems in a Branch & Cut scheme. Quesada and Grossmann [46] first proposed a method, called LP/NLP B&B, that combines the outer approximation framework with B&B, resulting in only a single dynamically updated B&B search tree. Such algorithms are referred to as single-tree methods, and several single-tree variants of both OA and ESH have been presented [11, 41, 42] and has been implemented by several MINLP solvers such as BONMIN [6], FilMint [1], AOA [26], SHOT [42], Pajarito [11], and BARON [29].

OA is regarded as one of the most efficient methods for convex MINLP [34] and several state-of-the-art solvers build upon the OA algorithm. However, as described in [36] methods such as ECP, ESH, and OA all similarly use the linear relaxation as in Kelley's cutting plane method [28]. Kelley's method is known to be unstable given its large jumps in the search space [24] and is generally considered to have a poor theoretical and practical performance, *e.g.,* see [44]. Several stabilization techniques [2, 12] have been proposed to tackle this shortcoming in the continuous problem setting (NLP). Directly using a trust-region or a regularization is non-trivial for mixed-integer problems due to the search space's discrete and often disjunct nature. For nonsmooth convex MINLP de Oliveira [45] proposed a regularized algorithm based on the ECP method. Combining OA and bundle methods, Delfino and de Oliveira [15] derived a method for nonsmooth convex MINLP. Kronqvist et al [36] showed that using ideas from the level method [30, 38] makes it possible to integrate regularization and second-order derivatives in an OA framework efficiently. By Using a second-order Taylor expansion of the Lagrangean within a level-based OA, the so-called Q-OA method [36] significantly reduced the number of iterations for highly nonlinear convex MINLP problems.

In this paper, we build upon the work by Kronqvist et al [36] and present a general regularization framework for OA. We refer to the new method as Regularized Outer-Approximation (ROA), which enables different regularization functions to be used while ensuring global convergence. We propose a set of regularization functions based on distance metrics and the Lagrangean. The motivation behind the Lagrangean-based regularization functions is to incorporate more information from both the objective and constraint function. We also integrate the regularization framework with the single-tree search algorithm in a method we denominate as regularized LP/NLP (RLP/NLP).

1.1 Contributions and outline

We propose a general framework for integrating different regularized mixed-integer subproblems in the OA method in multi-tree and single-tree settings to solve convex MINLP problems. We prove that these methods are guaranteed to converge to the optimal solution of MINLP problems, regardless of the choice of regularization function. Seven different regularization functions are proposed as objectives in this work, three of them coming from distance metrics to the incumbent solutions. The other four have approximations of the Lagrangean function around the best-found solution. We implemented these methods in the open-source **M**ixed-**i**nteger **n**onlinear **d**ecomposition **t**oolbox for **Py**omo - MindtPy [3]. The implementation is used for a comprenhensive computational study by solving all convex MINLP problems available in the benchmark library MINLPLib [9].

The remaining manuscript is organized as follows. In Section 2 we provide a brief background on the OA and LP/NLP methods. Section 3 introduces the Regularized Outer-Approximation (ROA) method and proposes the norm-based objective functions for the regularization subproblem. Next, we introduce four objective functions obtained through approximations of the Lagrangean function in Section 4. We provide a convergence analysis of the proposed methods in Section 5. The single-tree extension of the regularization method as the Regularization LP/NLP Branch & Bound (RLP/NLP) method and its implementation are presented in Section 6. Finally, the computational results of the methods' benchmarking in presented in Section 7.

2 Background

The MINLP problems considered in this paper are of the form,

$$\begin{aligned} \min_{\mathbf{x},\mathbf{y}} \quad & f(\mathbf{x},\mathbf{y}) \\ \text{s.t.} \quad & g_j(\mathbf{x},\mathbf{y}) \leqslant 0 \quad \forall j = 1,\dots,l, \\ & \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} \leqslant \mathbf{b}, \\ & \mathbf{x} \in \mathbb{R}^n,\ \mathbf{y} \in \mathbb{Z}^m. \end{aligned} \tag{MINLP}$$

Later in the algorithms, the (nonlinear) objective function is transformed into a constraint by the epigraph formulation, $f(\mathbf{x},\mathbf{y}) \leqslant \mu$, where the continuous variable μ represents the objective value. To guarantee global convergence for OA-type algorithms typically require convexity assumptions, a bounded search space, and some form of constraint qualification for problem MINLP [6, 16, 18]. Throughout this paper, we rely on the following assumptions:

Assumption 1. The nonlinear functions $f, g_1, \dots, g_l : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ are convex and continuously differentiable.

Assumption 2. The linear constraints $\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} \leqslant \mathbf{b}$ form a bounded polyhedron.

Assumption 3. A constraint qualification holds for each feasible integer combination, *e.g.,* Slater's condition [48].

We will not go through the Outer-Approximation method in detail. However, we will introduce some of the main concepts and subproblems as these are used later on. Algorithm 3 in the Appendix summarizes the main steps of the OA algorithm, and for more details, we refer to [16, 18, 36, 46].

The OA method uses a linear approximation (or relaxation) of the feasible set of the nonlinear constraints. Given a set of trial solutions $\left\{(\mathbf{x}^i,\mathbf{y}^i)\right\}_{i=0}^k$, the linear relaxation is formed by the constraints

$$\begin{aligned} f(\mathbf{x}^k,\mathbf{y}^k) + \nabla f(\mathbf{x}^k,\mathbf{y}^k)^\top \begin{bmatrix} \mathbf{x}-\mathbf{x}^k \\ \mathbf{y}-\mathbf{y}^k \end{bmatrix} &\leqslant \mu, \\ g_j(\mathbf{x}^k,\mathbf{y}^k) + \nabla g_j(\mathbf{x}^k,\mathbf{y}^k)^\top \begin{bmatrix} \mathbf{x}-\mathbf{x}^k \\ \mathbf{y}-\mathbf{y}^k \end{bmatrix} &\leqslant 0 \quad \forall j \in \mathcal{I}_k, \end{aligned} \tag{1}$$

where $\mathcal{I}_i$ are index sets containing the indices of the nonlinear constraint active at the trial solution $(\mathbf{x}^i,\mathbf{y}^i)$ [18]. The constraints in (1) form a polyhedral outer approximation of the feasible set of the nonlinear constraints in problem MINLP. The linear constraints are often referred to as cuts, as they refine the outer approximation by cutting off infeasible parts of the search space.

In the OA algorithm the next integer combination $\mathbf{y}^{k+1}$ is obtained by solving the following MILP problem

$$\begin{aligned} \min_{\mathbf{x},\mathbf{y},\mu} \quad & \mu \\ \text{s.t.} \quad & f(\mathbf{x}^i,\mathbf{y}^i) + \nabla f(\mathbf{x}^i,\mathbf{y}^i)^\top \begin{bmatrix} \mathbf{x}-\mathbf{x}^i \\ \mathbf{y}-\mathbf{y}^i \end{bmatrix} \leqslant \mu \quad \forall i = 1,\dots,k, \\ & g_j(\mathbf{x}^i,\mathbf{y}^i) + \nabla g_j(\mathbf{x}^i,\mathbf{y}^i)^\top \begin{bmatrix} \mathbf{x}-\mathbf{x}^i \\ \mathbf{y}-\mathbf{y}^i \end{bmatrix} \leqslant 0 \quad \forall i = 1,\dots k, \forall j \in \mathcal{I}_i, \\ & \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} \leqslant \mathbf{b}, \\ & \mathbf{x} \in \mathbb{R}^n,\ \mathbf{y} \in \mathbb{Z}^m, \mu \in \mathbb{R}. \end{aligned} \tag{OA-MILP}$$

Due to the convexity assumption, we know that the optimum of problem OA-MILP provides a valid lower bound (LB) to the MINLP problem, referred to as LB^{k+1}. If the the integer assignment $\mathbf{y}^{k+1}$ is feasible, then the corresponding continuous variables $\mathbf{x}^{k+1}$ are determined by solving the convex NLP subproblem

$$\begin{aligned} \min_{\mathbf{x}\in\mathbb{R}^n} \quad & f(\mathbf{x},\mathbf{y}^{k+1}) \\ \text{s.t.} \quad & g_j(\mathbf{x},\mathbf{y}^{k+1}) \leqslant 0 \quad \forall j = 1,\dots l, \\ & \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y}^{k+1} \leqslant \mathbf{b}. \end{aligned} \tag{NLP-I}$$

A feasible solution to problem NLP-I gives a valid upper bound (UB) to the MINLP problem, which we refer to as UB^{k+1}.

If problem NLP-I is infeasible, then values for the continuous variables $\mathbf{x}^{k+1}$ are obtained by solving a feasibility problem. The feasibility problem minimizes the norm of the constraint violations, typically ℓ_∞ or ℓ_1. With the current choice of integer variables $\mathbf{y}$, the feasibility problem is defined as

$$\begin{aligned} \min_{\mathbf{x}\in\mathbb{R}^n,\mathbf{s}\in\mathbb{R}^l_+} \quad & \|\mathbf{s}\|_p \\ \text{s.t.} \quad & g_j(\mathbf{x},\mathbf{y}^{k+1}) \leqslant s_j \quad \forall j = 1,\dots l, \\ & \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y}^{k+1} \leqslant \mathbf{b}. \end{aligned} \tag{NLP-f}$$

As described in Algorithm 3, the OA method iteratively solves subproblems OA-MILP, NLP-I, and NLP-f to find the optimal solution to the MINLP problems and produce certificates of optimality (upper and lower bounds).

Every iteration of the OA algorithm solves a new problem OA-MILP that only differs from the previous one by some cuts. To avoid solving a large number of similar and potentially challenging MILP problems, Quesada and Grossmann [46] proposed the LP/NLP-based B&B algorithm that combines OA and B&B. The LP/NLP-based B&B algorithm dynamically updates problem OA-MILP and only builds a single B&B tree. Each node, or leaf, of the search tree forms a continuous linear programming (LP) problem where the integer variables are relaxed as continuous, and the cuts in (1) are used to approximate the nonlinear constraints. Integer solutions are obtained through

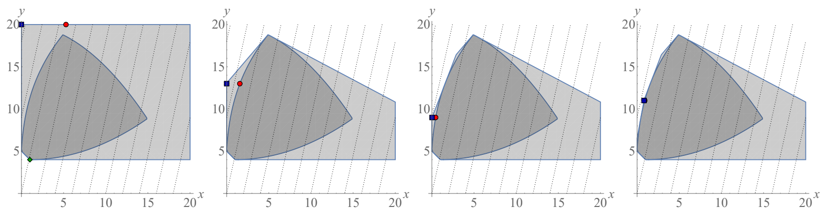
Fig. 1: Progress of OA in problem Ex 1, with each figure being an iteration. The feasible region defined by the nonlinear constraints (dark gray), the outer approximation obtained by the generated cuts (light gray), the MIP problem solution (■), and NLP subproblem solution (●) are included.

branching on the LP problems. Once an integer solution is found in the search tree, it is used as a new integer combination in the OA algorithm resulting in new cuts by solving the corresponding NLP subproblem. The best-found feasible solution to the original problem is known as the incumbent solution and is used as an upper bound in the search tree. The linear B&B procedure continues with an improved approximation of the nonlinear constraints. The new cuts, derived from the new integer combination, are added to all open nodes of the B&B tree. The main steps in the LP/NLP B&B are outlined in Algorithm 4 in the Appendix.

We consider the following illustrative example to highlight the features of the presented methods and show how they differ from OA.

$$\begin{aligned} \text{minimize} \quad & x - y/4.5 + 2 \\ \text{s.t.} \quad & x^2/20 + y \leqslant 20 \\ & (x-1)^2/40 - y \leqslant -4 \\ & 0.275y^{1.5} - 10(x+0.1)^{0.5} \leqslant 0 \\ & 0 \leqslant x \leqslant 20, \quad 0 \leqslant y \leqslant 20, \quad x \in \mathbb{R},\ y \in \mathbb{Z}. \end{aligned} \tag{Ex 1}$$

To best compare all the methods, we use the feasible point $(x^0, y^0) = (1, 4)$ as the starting point. OA requires five iterations to solve this problem, of which the first four iterations are shown in Figure 1. In this specific problem, the first iteration results in an infeasible solution. The optimal solution is obtained in iteration four, and verifying optimality requires an additional iteration.

3 Regularized Outer-Approximation

The level-based OA (L-OA) method was presented by Kronqvist et al [36], where the authors used a squared ℓ_2-regularization to the subproblem of obtaining new integer assignments. It was shown in the paper that the regularization technique is equivalent to adding a trust region, given by squared ℓ_2-norm, with a center at the incumbent solution. We give a brief overview of the L-OA algorithm since the other regularization techniques in this paper are also based on this framework. For more details, we refer to [36].

At iteration k, the problem OA-MILP is solved to obtain a LB LB^k on the optimal objective value. Given an incumbent solution $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$ and the UB resulting from $f(\bar{\mathbf{x}}, \bar{\mathbf{y}})$, we estimate the optimal objective value of the MINLP problem $f^\star = f(\mathbf{x}^\star, \mathbf{y}^\star)$ as

$$\hat{f}_k^\star = (1-\alpha) f(\bar{\mathbf{x}}, \bar{\mathbf{y}}) + \alpha LB^k, \tag{2}$$

where $\alpha \in (0,1]$. The estimated optimum $\hat{f}_k^\star$ is chosen as an interpolation between the UB and LB, where α is the interpolation parameter representing how much the linear approximation is trusted. For the continuous setting, within the level method proposed by Lemaréchal et al [38], a value of $\alpha = 1 - \sqrt{2}/2 \approx 0.29$ is found to be optimal. The proof does not generalize for the mixed-integer case, meaning that an ideal value for α is not known a-priori. As in [36], we also simply use $\alpha = 0.5$. The next integer assignment $\mathbf{y}^{k+1}$ is now determined by projecting $\bar{\mathbf{x}}, \bar{\mathbf{y}}$ onto the $\hat{f}_k^\star$ level set of the linearly approximated objective function intersected with the current outer approximation of the feasible set. The projected solution is obtained as the minimizer of the following MIP problem,

$$\begin{aligned} \min_{\mathbf{x},\mathbf{y},\mu} \quad & \phi^h_{\bar{\mathbf{x}},\bar{\mathbf{y}}}(\mathbf{x},\mathbf{y}) \\ \text{s.t.} \quad & \mu \leqslant \hat{f}_k^\star \\ & f(\mathbf{x}^i,\mathbf{y}^i) + \nabla f(\mathbf{x}^i,\mathbf{y}^i)^\top \begin{bmatrix} \mathbf{x}-\mathbf{x}^i \\ \mathbf{y}-\mathbf{y}^i \end{bmatrix} \leqslant \mu \quad \forall i = 1,\dots,k, \\ & g_j(\mathbf{x}^i,\mathbf{y}^i) + \nabla g_j(\mathbf{x}^i,\mathbf{y}^i)^\top \begin{bmatrix} \mathbf{x}-\mathbf{x}^i \\ \mathbf{y}-\mathbf{y}^i \end{bmatrix} \leqslant 0 \quad \forall i = 1,\dots,k, \forall j \in \mathcal{I}_i, \\ & \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} \leqslant \mathbf{b}, \\ & \mathbf{x} \in \mathbb{R}^n,\ \mathbf{y} \in \mathbb{Z}^m, \mu \in \mathbb{R}, \end{aligned} \tag{MIP-Proj}$$

where $\phi^h_{\bar{\mathbf{x}},\bar{\mathbf{y}}} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ is a convex regularization function represent by the symbol h. The parameter α value is bounded between 0 and 1. It represents the trade-off between trusting the solution of the linear relaxation of the problem ($\alpha = 1$), which leads to an optimal solution of problem MIP-Proj with the same minimizer of problem OA-MILP, and staying in the neighborhood of problem NLP-I solution ($\alpha \to 0$). The L-OA algorithm in [36] use the regularization function

$$\phi^{\ell_2^2}_{\bar{\mathbf{x}},\bar{\mathbf{y}}}(\mathbf{x},\mathbf{y}) := \left\| \begin{matrix} \mathbf{x}-\bar{\mathbf{x}} \\ \mathbf{y}-\bar{\mathbf{y}} \end{matrix} \right\|_2^2, \tag{3}$$

and the authors mention that the convergence guarantees of the algorithm are independent of the choice of objective function in MIP-Proj. The regularization problem MIP-Proj must contain all the cuts accumulated in problem OA-MILP to ensure convergence. The regularization role is to favor solutions close to the incumbent solution concerning a specific metric. The new integer assignment $\mathbf{y}^{k+1}$ is chosen as a point as close as possible to the incumbent solution, such that the linearly approximated objective is reduced to at most $\hat{f}_k^\star$. By construction, the regularization problem MIP-Proj is always feasible, *e.g.,* the minimizer of problem OA-MILP will satisfy all the constraints, and it is used to derive the next integer assignment $\mathbf{y}^{k+1}$. Once the new integer combination is obtained, the corresponding continuous variables can be determined using the same technique as the OA method. The difference between the L-OA and OA methods is how the new integer assignments are obtained. Otherwise, both methods use the same techniques for determining the continuous variables and improving the outer approximation of the feasible set.

Since finite convergence of L-OA holds for any objective function in the regularization problem [36], other regularization techniques can easily be incorporated into the L-OA framework. A general framework based on the L-OA concept, where the regularization function is not specified, is summarized as a pseudo-code in Algorithm 1. We refer to this algorithm as Regularized Outer-Approximation (ROA).

Two alternative regularization functions that fits directly into the L-OA are

$$\phi^{\ell_1}_{\bar{\mathbf{x}},\bar{\mathbf{y}}}(\mathbf{x},\mathbf{y}) := \left\| \begin{matrix} \mathbf{x}-\bar{\mathbf{x}} \\ \mathbf{y}-\bar{\mathbf{y}} \end{matrix} \right\|_1 , \tag{4}$$

$$\phi^{\ell_\infty}_{\bar{\mathbf{x}},\bar{\mathbf{y}}}(\mathbf{x},\mathbf{y}) := \left\| \begin{matrix} \mathbf{x}-\bar{\mathbf{x}} \\ \mathbf{y}-\bar{\mathbf{y}} \end{matrix} \right\|_\infty . \tag{5}$$

A benefit of using a regularization based on either the ℓ_1-norm or ℓ_∞-norm is that the regularization problem can be encoded as a MILP problem. We define for the remaining of the paper the L-OA approach from [36] as ROA-ℓ_2^2, and the proposed linear regularization approaches that use (4) and (5) as regularization functions as ROA-ℓ_1 and ROA-ℓ_∞, respectively.

As shown in [36], L-OA finds the same integer solutions as problem OA-MILP in OA with specific trust-region constraints. In fact, the equivalence to a trust region still holds with the regularization given by any p-norm. This property is stated in Theorem 1. The proof uses the same argumentation as in [36] but is included for the sake of completeness.

Theorem 1 *With the regularization given by a p-norm, the procedure of solving problems OA-MILP and MIP-Proj in ROA results in solution equivalent to adding the trust region constraint*

$$\left\| \begin{matrix} \mathbf{x}-\bar{\mathbf{x}} \\ \mathbf{y}-\bar{\mathbf{y}} \end{matrix} \right\|_p \leqslant r_k \tag{6}$$

to problem OA-MILP in OA, where r_k is chosen as the optimum of problem MIP-Proj.

Algorithm 1 An algorithm summarizing the Regularized Outer-Approximation (ROA) method.

Define accepted optimality gap $\epsilon \geqslant 0$, the regularization function $\phi^h_{\bar{\mathbf{x}},\bar{\mathbf{y}}}$, and choose the parameter $\alpha \in (0, 1]$.

```
1. Initialization.
   1.1 Obtain a relaxed solution $\tilde{\mathbf{x}}, \tilde{\mathbf{y}}$ by solving an integer relaxation of the MINLP problem.
   1.2 Generate cuts at $\tilde{\mathbf{x}}, \tilde{\mathbf{y}}$ according to (1) and construct problems OA-MILP.
   1.3 Set iteration counter $k = 1$, $UB^0 = \infty$ and $LB^0 = -\infty$.
2. Repeat until $UB^{k-1} - LB^{k-1} \leqslant \epsilon$.
   2.1 Solve problem OA-MILP to obtain $\mathbf{y}^k$ and $LB^k$.
   2.2 If a feasible solution $\bar{\mathbf{x}}, \bar{\mathbf{y}}$ has been found, calculate the estimated optimal value $\hat{f}^\star_k$ according to (2) and solve problem MIP-Proj to update $\mathbf{y}^k$.
   2.3 Solve problem NLP-I with integer variables fixed as $\mathbf{y}^k$ to obtain $\mathbf{x}^k$.
      2.3.1 If problem NLP-I is feasible, set $UB^k = \min\{f(\mathbf{x}^k, \mathbf{y}^k), UB^{k-1}\}$.
         2.3.1.1 If $f(\mathbf{x}^k, \mathbf{y}^k) \leqslant f(\bar{\mathbf{x}}, \bar{\mathbf{y}})$, set $\bar{\mathbf{x}}, \bar{\mathbf{y}} = \mathbf{x}^k, \mathbf{y}^k$.
      2.3.2 If problem NLP-I is infeasible, obtain $\mathbf{x}^k$ by solving feasibility problem NLP-f and set $UB^k = UB^{k-1}$.
   2.4 Generate cuts at $\mathbf{x}^k, \mathbf{y}^k$ according to (1) and add these to problems OA-MILP and MIP-Proj.
   2.5 (Optional) Generate no-good cuts at $\mathbf{y}^k$ and add these to problems OA-MILP.
   2.5 Increase iteration counter, $k = k + 1$,
3. Return $\bar{\mathbf{x}}, \bar{\mathbf{y}}$ as the optimal solution $\mathbf{x}^\star, \mathbf{y}^\star$.
```

Proof As mentioned earlier, MIP-Proj is always feasible and and we denote the minimizer by $\mathbf{x}^{\text{MIP-Proj}}, \mathbf{y}^{\text{MIP-Proj}}, \mu^{\text{MIP-Proj}}$. The radius of the equivalent trust region constraint is then given by

$$r_k = \left\| \begin{matrix} \mathbf{x}^{\text{MIP-Proj}} - \bar{\mathbf{x}} \\ \mathbf{y}^{\text{MIP-Proj}} - \bar{\mathbf{y}} \end{matrix} \right\|_p \tag{7}$$

Solving problem OA-MILP, with the trust region constraint, gives the solution $\mathbf{x}^{\text{MILP}}, \mathbf{y}^{\text{MILP}}, \mu^{\text{MILP}}$. Now, assume this solution is not an optimal solution to problem MIP-Proj. Since $\mathbf{x}^{\text{MILP}}, \mathbf{y}^{\text{MILP}}, \mu^{\text{MILP}}$ is not an optimal solution, it follows that

$$r_k > \left\| \begin{matrix} \mathbf{x}^{\text{MILP}} - \bar{\mathbf{x}} \\ \mathbf{y}^{\text{MILP}} - \bar{\mathbf{y}} \end{matrix} \right\|_p . \tag{8}$$

Since OA-MILP minimizes μ, we know that $\mu^{\text{MILP}} \leqslant \mu^{\text{MIP-Proj}} \leqslant \hat{f}^\star_k$. This leads to a contradiction since $\mathbf{x}^{\text{MILP}}, \mathbf{y}^{\text{MILP}}, \mu^{\text{MILP}}$ is a feasible solution to problem MIP-Proj with an objective value strictly lower than the solution obtained by solving the minimization problem. Therefore, the solution to problem OA-MILP, with the trust-region constraint, must also be an optimal solution to problem MIP-Proj. □

Depending on which function $\phi^h_{\bar{\mathbf{x}},\bar{\mathbf{y}}}$ is used in the ROA method, we obtain different variants of the algorithm. These variants are denoted as ROA-h, *e.g.,* we refer to ROA-ℓ_1 when (4) is used as the objective for the regularization problem. Next, we illustrate the difference between these variants with example Ex 1.

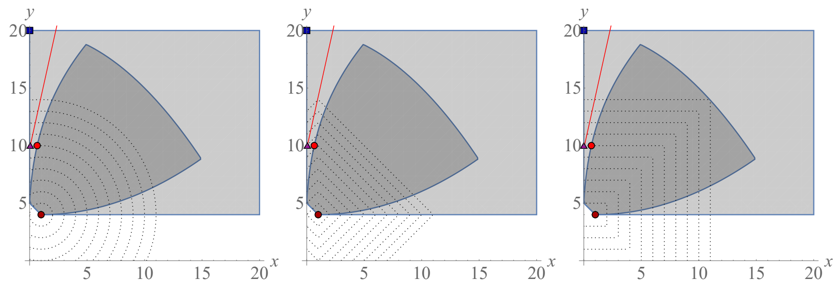

Fig. 2: First iteration of ROA for problem Ex 1 with the three level norms presented in this work. The format from Figure 1 is used here, with the additional features that the regularization objective contours, the regularization problem solution (▲), the incumbent solution (●), and the level constraint (2) with $\alpha = 0.5$ (red line) are included. Left: ROA-ℓ_2^2. Center: ROA-ℓ_1. Right: ROA-ℓ_∞.

For example Ex 1, the three level regularization norms presented here; ROA-ℓ_2^2, ROA-ℓ_1, and ROA-ℓ_∞; converge to the optimal solution in three iterations. Thanks to the regularization, all three approaches find the optimal solution in the first iteration, as observed in Figure 2. Although the simple example does not show this behavior, the regularization objective's choice might affect which integer combination gets chosen to solve problem NLP-I. In every case, the regularization keeps this integer combination close to the incumbent solution. Moreover, the choice of the objective may impact the computational time required to solve the regularization problem. As mentioned above, choosing the squared ℓ_2 norm as in L-OA [36] leads to the regularization problem becoming an MIQP. On the other hand, the ℓ_1 and ℓ_∞ norms in the objectives can be modeled using linear inequalities and auxiliary variables, as presented in the Appendix, leading to MILP regularization subproblems. For all approaches, it takes two more iterations to close the LB.

In the next section, we present two new regularization strategies that also fit within the ROA framework and incorporate information from the Lagrangean function.

4 Lagrangean based regularization

To take advantage of second-order derivatives for selecting the new integer assignment $\mathbf{y}^{k+1}$, Kronqvist et al [36] proposed a technique they refer to as Quadratic Outer-Approximation (Q-OA). Instead of a regularization function, Q-OA uses a second-order Taylor series expansion of the Lagrangean function as the objective function in MIP-Proj. Thus, the new integer assignment is chosen by minimizing a quadratic approximation of the Lagrangean within an outer approximation of the feasible set subject to a level constraint, *i.e.*,

$\mu \leqslant \hat{f}_k^\star$. Except for the level constraint, there is an apparent similarity with the sequential quadratic programming (SQP) approach [5].

The Lagrangean function $\mathcal{L} : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^k \to \mathbb{R}$ associated with the MINLP problem can be written as

$$\mathcal{L}(\mathbf{x}, \mathbf{y}, \lambda) = f(\mathbf{x}, \mathbf{y}) + \lambda^\top \tilde{g}(\mathbf{x}, \mathbf{y}), \tag{9}$$

where $\tilde{g} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^k$ contains all the constraints in the form $\tilde{g}(\mathbf{x}, \mathbf{y}) \leqslant 0$, linear and nonlinear. From the fixed NLP problem giving the incumbent solution $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$, the corresponding dual variable $\bar{\lambda}$ is also obtained. Now, by defining the regularization function $\phi^h_{\bar{\mathbf{x}},\bar{\mathbf{y}}}$ as

$$\phi^{\mathcal{L}_2}_{\bar{\mathbf{x}},\bar{\mathbf{y}}}(\mathbf{x}, \mathbf{y}) := \nabla_{\mathbf{x},\mathbf{y}} \mathcal{L}(\bar{\mathbf{x}}, \bar{\mathbf{y}}, \bar{\lambda})^\top \begin{bmatrix} \mathbf{x} - \bar{\mathbf{x}} \\ \mathbf{y} - \bar{\mathbf{y}} \end{bmatrix} + \frac{1}{2} \begin{bmatrix} \mathbf{x} - \bar{\mathbf{x}} \\ \mathbf{y} - \bar{\mathbf{y}} \end{bmatrix}^\top \nabla^2_{\mathbf{x},\mathbf{y}} \mathcal{L}(\bar{\mathbf{x}}, \bar{\mathbf{y}}, \bar{\lambda}) \begin{bmatrix} \mathbf{x} - \bar{\mathbf{x}} \\ \mathbf{y} - \bar{\mathbf{y}} \end{bmatrix}, \tag{10}$$

the ROA method in Algorithm 1 will result in the Q-OA algorithm. Note that $\phi^h_{\bar{\mathbf{x}},\bar{\mathbf{y}}}$ in (10) can be considered a regularizer with a stabilization center at the minimizer of the quadratic approximation of the Lagrangean. Suppose the Hessian of the Lagrangean is not positive definite (only positive semidefinite). In that case, the stabilization center may not be a unique point but a subspace.

Remark 1 With the integer variables fixed as $\bar{\mathbf{y}}$, the point $(\bar{\mathbf{x}}, \bar{\mathbf{y}}, \bar{\lambda})$ is a stationary point of the Lagrangean and, therefore, all partial derivatives corresponding to the continuous variables will be zero in $\nabla_{\mathbf{x},\mathbf{y}} \mathcal{L}(\bar{\mathbf{x}}, \bar{\mathbf{y}}, \bar{\lambda})$. This follows directly from the KKT conditions of the NLP problem NLP-I.

Next, we derive two new regularization functions based on the Lagrangean that can be directly implemented in Algorithm 1. Using the Hessian of the Lagragian, we can define a norm as

$$\left\| \begin{matrix} \mathbf{x} \\ \mathbf{y} \end{matrix} \right\|_{\mathcal{L}(\bar{\mathbf{x}}, \bar{\mathbf{y}}, \bar{\lambda})} := \sqrt{\begin{bmatrix} \mathbf{x} - \bar{\mathbf{x}} \\ \mathbf{y} - \bar{\mathbf{y}} \end{bmatrix}^\top \nabla^2_{\mathbf{x},\mathbf{y}} \mathcal{L}(\bar{\mathbf{x}}, \bar{\mathbf{y}}, \bar{\lambda}) \begin{bmatrix} \mathbf{x} - \bar{\mathbf{x}} \\ \mathbf{y} - \bar{\mathbf{y}} \end{bmatrix}}, \tag{11}$$

which is a proper norm if in the Hessian is positive definite or a semi norm if the Hessian is positive semidefinite [31]. Based on this (semi) norm, we define a new regularization function as

$$\phi^{\nabla^2 \mathcal{L}}_{\bar{\mathbf{x}},\bar{\mathbf{y}}}(\mathbf{x}, \mathbf{y}) := \left\| \begin{matrix} \mathbf{x} \\ \mathbf{y} \end{matrix} \right\|^2_{\mathcal{L}(\bar{\mathbf{x}}, \bar{\mathbf{y}}, \bar{\lambda})} = \begin{bmatrix} \mathbf{x} - \bar{\mathbf{x}} \\ \mathbf{y} - \bar{\mathbf{y}} \end{bmatrix}^\top \nabla^2_{\mathbf{x},\mathbf{y}} \mathcal{L}(\bar{\mathbf{x}}, \bar{\mathbf{y}}, \bar{\lambda}) \begin{bmatrix} \mathbf{x} - \bar{\mathbf{x}} \\ \mathbf{y} - \bar{\mathbf{y}} \end{bmatrix}. \tag{12}$$

This regularization function's motivation favors search directions in which the Lagrangean has a locally linear behavior. This regularization, therefore, favors regions of the search space where the outer approximation is expected to be more accurate. Suppose the Hessian has at least one zero eigenvalue. In that case, the equivalent trust-region will be unbounded in directions in which the quadratic approximation of the Lagrangean changes linearly.

There are situations in which the Hessian is not known or too expensive to compute. One of the simplest approximations of the Hessian is a scaled identity matrix ρI. The Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm [17], for example, uses the scaled identity as the first estimate of the Hessian of the Lagrangean. Using this trivial approximation of the Hessian in the quadratic approximation of the Lagrangean, gives us the following regularization function

$$\phi_{\bar{\mathbf{x}},\bar{\mathbf{y}}}^{\mathcal{L}_1/\ell_2^2}(\mathbf{x},\mathbf{y}) := \nabla_{\mathbf{x},\mathbf{y}}\mathcal{L}(\bar{\mathbf{x}},\bar{\mathbf{y}},\bar{\lambda})^\top \begin{bmatrix}\mathbf{x}-\bar{\mathbf{x}}\\ \mathbf{y}-\bar{\mathbf{y}}\end{bmatrix} + \rho \begin{bmatrix}\mathbf{x}-\bar{\mathbf{x}}\\ \mathbf{y}-\bar{\mathbf{y}}\end{bmatrix}^\top I \begin{bmatrix}\mathbf{x}-\bar{\mathbf{x}}\\ \mathbf{y}-\bar{\mathbf{y}}\end{bmatrix}, \tag{13}$$

where $\rho \in \mathbb{R}_+$ is a scaling factor. This gives a regularization function with a stabilization center shifted in the direction of the negative gradient of the Lagrangean. Since the gradient is zero for all the continuous variables, the stabilization center is only shifted for the discrete variables. The stabilization center $(\mathbf{x}_c, \mathbf{y}_c)$ can easily be determined from the stationary conditions of the regularization function, and is given by

$$\begin{bmatrix}\mathbf{x}_c\\ \mathbf{y}_c\end{bmatrix} = \begin{bmatrix}\bar{\mathbf{x}}\\ \bar{\mathbf{y}}\end{bmatrix} - \frac{\nabla_{\mathbf{x},\mathbf{y}}\mathcal{L}(\bar{\mathbf{x}},\bar{\mathbf{y}},\bar{\lambda})}{2\rho}. \tag{14}$$

Depending on the magnitude of both $\nabla_{\mathbf{x},\mathbf{y}}\mathcal{L}(\bar{\mathbf{x}},\bar{\mathbf{y}},\bar{\lambda})$ and ρ, the stabilization center might be far from the incumbent solution and even outside of the variable bounds. However, we can directly control how far from the incumbent solution the stabilization center lies by scaling ρ. By selecting ρ as

$$\rho = \left\| \frac{\nabla_{\mathbf{x},\mathbf{y}}\mathcal{L}(\bar{\mathbf{x}},\bar{\mathbf{y}},\bar{\lambda})}{2d} \right\|_2, \tag{15}$$

the euclidean distance between the stabilization center and the incumbent solution becomes d. We can, thus, use the parameter d to determine how far the stabilization center is shifted.

If we completely remove the quadratic term from the Lagrangean approximation, we are left with the linear approximation function

$$\phi_{\bar{\mathbf{x}},\bar{\mathbf{y}}}^{\mathcal{L}_1}(\mathbf{x},\mathbf{y}) := \nabla_{\mathbf{x},\mathbf{y}}\mathcal{L}(\bar{\mathbf{x}},\bar{\mathbf{y}},\bar{\lambda})^\top \begin{bmatrix}\mathbf{x}-\bar{\mathbf{x}}\\ \mathbf{y}-\bar{\mathbf{y}}\end{bmatrix}. \tag{16}$$

Note that function (16) will not result in a regularization in ROA! However, since the linear approximation function combines the gradients of both the constraints and the objective, it could provide a direction more favorable for finding feasible solutions. Based on the computational results in Section 7, we observe that using (16) as the objective function in the regularization subproblem is not advantageous compared to the other approaches presented in this paper; supporting the use of a regularizer.

Similarly to the level-based approaches, we use the following notation for the regularization methods derived using the Lagrangean: The Q-OA method presented in [36] is presented as ROA-$\mathcal{L}_2$. We denote ROA-$\mathcal{L}_1$ the method using the first-order approximation of the Lagrangean as in (16), and following that notation the regularization methods involving (12) and (13) are denoted ROA-$\nabla^2\mathcal{L}$ and ROA-$\mathcal{L}_1/\ell_2^2$, respectively.

Next, we illustrate the differences between these regularization functions derived from the Lagrangean in ROA with example Ex 1. In Figure 3 we observe the first three iterations of the ROA methods with objective functions for problem MIP-Proj given by the second-order Taylor approximation, the Hessian of the Lagrangean based norm, and the first-order Taylor approximations of the Lagrangean function. Notice that the second-order Taylor approximation of the Lagrangean, proposed initially as Q-OA in [36], has a regularization objective equivalent to the sum of the two other methods presented in Figure 3. This can be observed as the contours of the regularization objective in the ROA-$\mathcal{L}_2$ method have a stabilization center (sometimes beyond the domain of the figure) specified by the ROA-$\nabla^2\mathcal{L}$ with a shift given by ROA-$\mathcal{L}_1$ corresponding objective in the direction of the discrete variable. This observation corresponds with Remark 1. Moreover, the gradient of the Lagrangean switches from pointing up or down depending on whether the incumbent solution is below or above the optimal solution, respectively. Although all the methods shown in Figure 3 can find the optimal solution following an infeasible first iteration, the number of required iterations to close the gap between UB and LB and guarantee optimality varies. It takes ROA-$\mathcal{L}_2$ five iterations, ROA-$\nabla^2\mathcal{L}$ six iterations, and ROA-$\mathcal{L}_1$ seven iterations to guarantee the optimality of the solution after finding the optimal solution in the last iteration for the first method and in the second-to-last iteration for the other two methods.

The progress of the ROA-$\mathcal{L}_1/\ell_2^2$ method is shown in Figure 4. This Figure exemplifies how the ℓ_2^2 norm stabilization center is shifted from the incumbent solution in the direction of the Lagrangean gradient. This distance of the shifting is given by parameter d, equal to one in this example. It can be seen from the smallest contour, representing a regularization objective of zero, on which the incumbent solution lies. In terms of the number of iterations, ROA-$\mathcal{L}_1/\ell_2^2$ is the most efficient method among all the ones presented here at solving problem Ex 1, finding the optimal solution on its first iteration and closing the gap between UB and LB in three iterations.

5 Convergence properties

The convergence proof of the L-OA algorithm presented in [36] is entirely independent of the objective function of the regularization problem MIP-Proj. Therefore, finite convergence of ROA, for any function $\phi^h_{\bar{\mathbf{x}},\bar{\mathbf{y}}}$, directly follows from the convergence proofs of L-OA. For completeness, we outline the main convergence property of ROA. For more details, we refer the reader to Section 5 in [36].

From the start, we assumed that all the nonlinear functions were convex (Assumption 1). This assumption is crucial since it ensures that the ROA methods' cuts do not cut off any feasible integer solution and that problem OA-

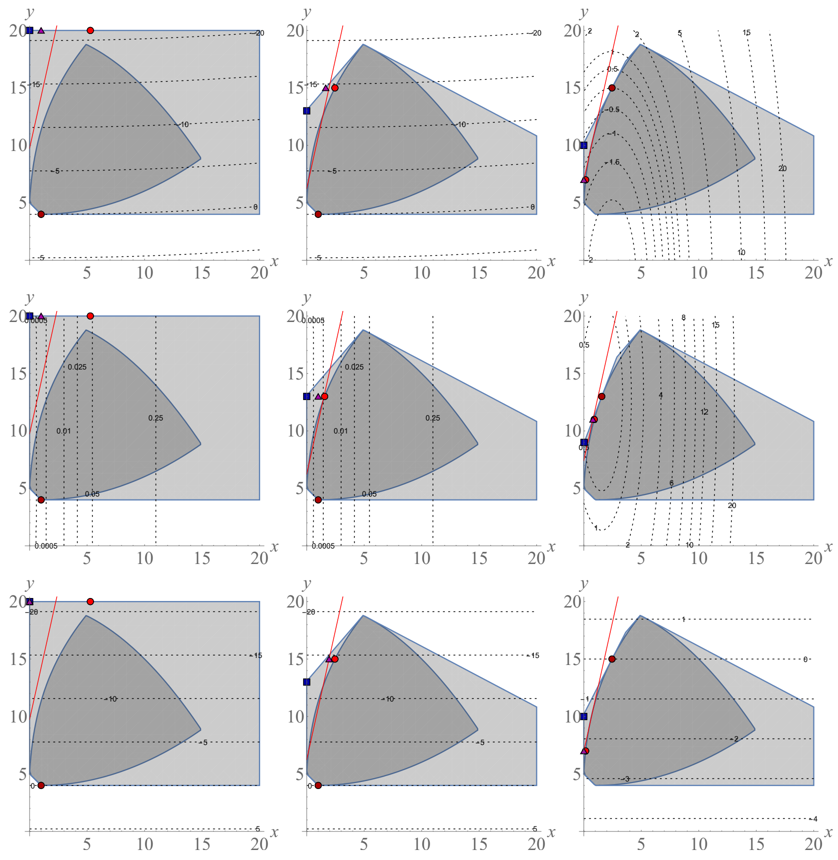

Fig. 3: First three iterations (from left to right) for proposed Lagrangean-based regularization methods for problem Ex 1. The format from Figure 2 is used here. Top: ROA-$\mathcal{L}_2$. Center: ROA-$\nabla^2\mathcal{L}$. Bottom: ROA-$\mathcal{L}_1$.

MILP gives a valid LB. For a complete proof that problem OA-MILP gives a valid LB, see [16, 18, 36]. The regularization problem will be feasible in each iteration with all the ROA methods. The feasibility of the regularization problem is given by the fact that the constraints in the regularization problem MIP-Proj are the same as in OA-MILP, besides the reduction constraint controlled by the confidence parameter α, for more details, see Lemma 4 in [36]. As stated in Lemma 3 in [36], it is clear that each infeasible integer combination obtained in the search will be excluded from the search space by the generated cuts. An essential property of the ROA methods is that, as long as the UB and LB of the optimal objective function are different, the regularization problem will provide a new integer combination in each iteration. This property is formally stated in the following theorem.

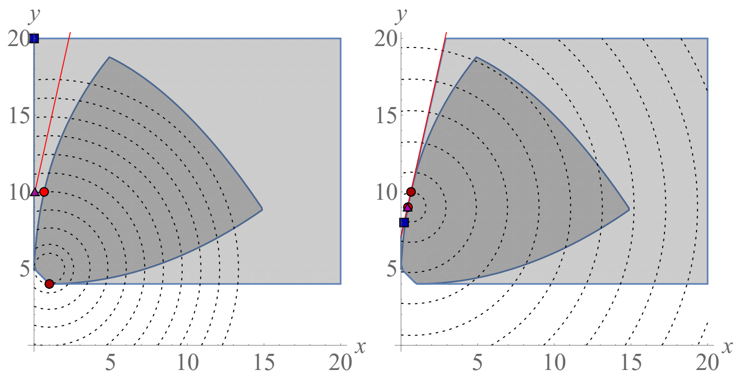

Fig. 4: First two iteration of ROA-$\mathcal{L}_1/\ell_2^2$ for problem Ex 1. The format from Figure 2 is used here, considering the scaling factor ρ such that the shifting of the stabilization center is $d = 1$.

Theorem 2 *If the lower bound is not equal to the upper bound, then the minimizer of regularization subproblem MIP-Proj provides a new integer combination.*

Proof By Lemma 3 in [36], it is clear that each infeasible integer combination encountered will be excluded from the search space by the cuts generated in the ROA algorithm. As proven in Theorem 5 in [36], all feasible integer combinations found by the ROA algorithm will also be excluded from the search space as long as there is a gap between the upper and lower bound. □

The main convergence property of ROA is summarized in the following theorem.

Theorem 3 *The ROA algorithm will terminate after a finite number of iterations, either by proving the best-found solution's optimality or by verifying that the MINLP problem is infeasible.*

Proof By Theorem 2, it is clear that problem MIP-Proj will, in each iteration, find a new, previously unexplored integer assignment, as long as the UB is not equal to the LB. As stated in Lemma 1 in [36], the LB is valid in each iteration of the algorithm. Due to Assumption 2, the search space only contains a finite number of different integer assignments. Therefore, the algorithm must terminate after a finite number of iterations with either the UB equal to the LB or by proving infeasibility by problem OA-MILP being infeasible. □

For more details and a complete convergence proof, we refer the reader to Section 5 in [36].

6 Regularization in LP/NLP Branch & Bound algorithm

Solvers based on a single-tree search or LP/NLP-based B&B algorithms have shown outstanding performance in recent benchmarks [34, 42]. A natural extension of the regularization framework from the previous section is, thus, the integration of regularization in LP/NLP-based B&B. This is also suggested in the conclusions and future work section of [36].

To introduce regularization into the LP/NLP-based B&B framework, we use the regularization problem MIP-Proj for each node in the search tree where an integer feasible solution is found. The regularization problem intends to choose new integer combinations close to the incumbent solution. It is also necessary to generate cuts at the nodes' variable values in the search tree with integer feasible solutions to ensure convergence. Otherwise, an infeasible integer combination encountered in the search tree might not be excluded as the regularization might result in a different integer combination. Except for these two modifications, the algorithm follows the same procedure as the standard LP/NLP-based B&B algorithm. The regularized LP/NLP-based B&B algorithm is summarized as a pseudo-code in Algorithm 2. We denote all the algorithms implementing regularization approaches on the LP/NLP-based B&B method as RLP/NLP. Similarly to ROA, depending on the objective function used in the regularization subproblem, $\phi^h_{\bar{\mathbf{x}},\bar{\mathbf{y}}}(\mathbf{x},\mathbf{y})$, we denominate the approach as RLP/NLP-h, *e.g.,* the single-tree approach using (5) is denoted RLP/NLP-ℓ_∞.

A significant difference compared to ROA is that the optimum of the linear approximation OA-MILP is not available during the search. This is an important detail and is described further in the following remark.

Remark 2 During the B&B tree search, the minimum of the current linear approximation OA-MILP is not known. Only a lower bound to problem OA-MILP is available, and all integer feasible solutions to OA-MILP may have a larger objective function value. Using the available LB to calculate the estimated optimum $\hat{f}^\star_k$ may, therefore, result in an infeasible level constraint, *i.e.,* there does not exist a feasible integer solution to problem OA-MILP with an objective value less than or equal to $\hat{f}^\star_k$. In such a situation, the regularization problem MIP-Proj will also be infeasible.

If the regularization is infeasible, the RLP/NLP algorithm continues by using the integer combination obtained at the current node. Note that each integer feasible node of the search tree involves a possibly expensive regularization problem. Therefore, the additional regularization problem must significantly reduce the number of integer combinations explored to be competitive. As shown in [36] the regularization can lead to a drastic reduction of iterations and explored integer combinations.

Since $\hat{f}^\star_k$ is not necessarily a valid LB to the current linear approximation OA-MILP, the convergence proofs from the previous section do not hold. However, the convergence can still be guaranteed as cuts are generated for each node's variable values with an integer feasible solution. Therefore, the conver-

Algorithm 2 An algorithm summarizing the regularized LP/NLP (RLP/NLP) method.

```
Define accepted optimality gap $\epsilon \geqslant 0$ and choose the parameter $\alpha \in (0,1]$.
1. Initialization.
  1.1 Obtain a relaxed solution $\tilde{\mathbf{x}}, \tilde{\mathbf{y}}$ by solving an integer relaxation of the MINLP problem.
  1.2 Generate cuts at $\tilde{\mathbf{x}}, \tilde{\mathbf{y}}$ according to (1) and construct problems OA-MILP.
  1.3 Set node counter $k = 1$, $UB^0 = \infty$ and $LB^0 = -\infty$.
2. Begin B&B search for problem OA-MILP and continue until $UB^{k-1} - LB^{k-1} \leqslant \epsilon$.
  2.1 If a new solution $\hat{\mathbf{x}}, \hat{\mathbf{y}}$ is found, then generate cuts at $\hat{\mathbf{x}}, \hat{\mathbf{y}}$, set $\mathbf{y}^k = \hat{\mathbf{y}}$ and update $LB^k$ according to current B&B tree. Add the new cuts to open nodes of the B&B tree and to problem MIP-Proj.
  2.2 If a feasible solution has been found or provided
    2.2.1 Calculate the estimated optimal value $\hat{f}_k^\star$ according to (2).
    2.2.2 Solve problem MIP-Proj to update $\mathbf{y}^k$. If the regularization problem is infeasible, keep $\mathbf{y}^k$ unchanged.
  2.3 Solve problem NLP-I with integer variables fixed as $\mathbf{y}^k$ to obtain $\mathbf{x}^k$ and $\lambda^{\mathbf{k}}$.
    2.3.1 If problem NLP-I is feasible, set $UB^k = \min\{f(\mathbf{x}^k, \mathbf{y}^k), UB^{k-1}\}$.
      2.3.1.1 If $f(\mathbf{x}^k, \mathbf{y}^k) \leqslant f(\bar{\mathbf{x}}, \bar{\mathbf{y}})$, set $\bar{\mathbf{x}}, \bar{\mathbf{y}}, \bar{\lambda} = \mathbf{x}^k, \mathbf{y}^k, \lambda^k$.
    2.3.2 If problem NLP-I is infeasible, obtain $\mathbf{x}^k$ by solving feasibility problem NLP-f.
  2.4 Generate cuts at $\mathbf{x}^k, \mathbf{y}^k$ according to (1) and add these to problem MIP-Proj, and add these as global lazy constraints to the B&B tree.
  2.5 (Optional) Generate no-good cuts at $\mathbf{y}^k$ and add these as global lazy constraints to the B&B tree of problem OA-MILP.
  2.6 Increase node counter, $k = k + 1$
3 Return $\bar{\mathbf{x}}, \bar{\mathbf{y}}$ as the optimal solution $\mathbf{x}^\star, \mathbf{y}^\star$.
```

gence is guaranteed due to the convergence of the ECP algorithm, see [1] for details of a single-tree ECP algorithm.

7 Computational results

This section introduces our implementation details of the seven ROA methods and analyzes their performance through benchmark tests. The OA method is selected as the baseline. As a general observation, we notice that the regularization methods can handle highly nonlinear convex MINLP problems more efficiently than OA. This aligns well with the observations of the two regularization methods in [36]. Using regularization methods induces a more careful choice of the integer combination to be evaluated. Having that trial solution lies preferably close to the best found feasible solution. In general, these regularization methods favor the choice of the following integer combination close to a stabilization center. This center is constructed using the incumbent solution and the constraints curvature, using information from the Lagrangean of the problem or a p-norm. The choice of the new integer solution comes at the expense of solving a mixed-integer regularization subproblem. This extra step might become exorbitant for mostly linear instances, where tight outer approximations of the nonlinear feasible region can be obtained with only a few gradient-based cuts.

To test the performance of the proposed methods, we use test instances from the problem library MINLPLib[1] [9]. There are 536 convex problems in MINLPLib, from which we select the 438 instances that have at least one discrete variable and at least one continuous variable. We denote the 438-instances set as Problem Set 1. Compared to the OA method, ROA methods use regularization to keep the trial solutions close to the incumbent solution and the feasible set. Favoring solutions close to the incumbent also favors areas close to a linearization point, *i.e.,* areas where the outer approximation is accurate. In theory, regularization-based methods lead to a higher efficiency gain concerning OA in highly nonlinear instances. This has been corroborated experimentally [36]. Therefore, we establish Problem Set 2 with highly nonlinear instances selected from Problem Set 1 according to the following criterion:

$$\frac{n_{nonlin}}{n+m} > 0.4, \tag{17}$$

where n_{nonlin} is the number of variables present in some nonlinear term, and $m+n$ is the total number of discrete and continuous variables. There are in total 135 convex MINLP problems in MINLPLib that satisfy (17). The instances in Problem Set 1 have between 2 to 4530 variables and 0 to 5329 constraints. The instances in Problem Set 2 range from 6 to 4530 variables and 0 to 4650 constraints.

7.1 Implementation details

The OA methods and seven ROA methods are implemented as part of the **M**ixed-**i**nteger **n**onlinear **d**ecomposition **t**oolbox for **Py**omo - MindtPy [3]. This toolbox presents an open-source[2] implementation of several solution techniques for MINLP based on problem decomposition. Through a Python implementation relying on the algebraic modeling language Pyomo [23], MindtPy can easily access a wide range of solvers to address the subproblems arising from the decomposition. The methods implemented in MindtPy for the solution of convex MINLP include OA [16] and ECP [53]. These are complemented with other decomposition methods such as the feasibility pump [4, 7] and the center cut algorithm [35]. Besides, MindtPy includes an implementation of the LB/NLP B&B method [46]. Its flexible framework allows users to easily tailor the algorithm to fit their particular application *i.e.,* by using different initialization procedures, feasibility norms, cutting planes generators, and call-back procedures.

For the results presented herein, we use CPLEX 20.1.0.0 [27] as the solver for the MILP/MIQP subproblems and IPOPT 3.12 [52] for the NLP subproblems using the Harwell Subroutine Library (HSL) MA27 [25] as a solver for linear systems. The level parameter in ROA methods is set to $\alpha = 0.5$ for all

[1] Retrieved on May 7, 2021, from `http://www.minlplib.org/`

[2] `https://pyomo.readthedocs.io/en/stable/contributed_packages/mindtpy.html`

approaches. Moreover, for ROA-$\mathcal{L}_1/\ell_2^2$ we use as shifting radius $d = 1$. We implement the multi-tree and single-tree approaches, described in Algorithms 1 and 2 respectively. We consider the zero tolerance for checking if a constraint is active as 10^{-8}; hence we set the IPOPT parameter `constr_viol_tol` to that value.

According to Theorem 3, it is not necessary to solve the regularization subproblems to optimality. As noted in [36], any feasible solution for the regularization problems is sufficient to guarantee the convergence of the ROA methods. These regularization problems are of the same size as the master OA-MILP, and solving them might be a limiting factor in ROA, as observed by [36] previously. The performance was improved by not solving the regularization problem to optimality by using the setting the MIP solution limit parameter, **`mip limits solutions`**, to 10. If CPLEX uses multiple threads, the number of MIP solutions found by CPLEX might be slightly greater than the **`mip limits solutions`** given that if the limit is reached, the nodes being processed in other threads will not be interrupted. CPLEX will stop after all the current working threads are completed.

Since the problems we consider are all convex, the Hessian of the Lagrangean is always positive semidefinite, and the regularization subproblems are always convex. However, due to numerical accuracy, the regularization problem ended up nonconvex for a few cases, *e.g.,* the smallest eigenvalue of the Hessian was slightly negative. Therefore, we set the `optimalitytarget` parameter to 3 to enable CPLEX to solve nonconvex MIQPs in the ROA-$\mathcal{L}_2$ and ROA-$\nabla^2\mathcal{L}$ methods. Another approach to deal with the nonconvexities induced by numerical accuracy is to add small perturbations to the diagonal of the Hessian [36].

The solution procedure is initialized by solving the continuous relaxation of problem MINLP, which provides a valid LB of the optimal objective function. In each iteration k, the problem OA-MILP is initialized with the NLP subproblem solution in iteration $k-1$. Since the solution of OA-MILP is feasible to the problem MIP-Proj, we use its optimal value to initialize the regularization subproblems. Along this line, as problem NLP-I is solved for the integer combination of the regularization problem, we use the solution to MIP-Proj to initialize the nonlinear subproblems. All the other settings in MindtPy, CPLEX, and IPOPT are the same as the default.

As termination criteria, we use the standard criteria of both an absolute optimality tolerance ϵ and a relative optimality tolerance ϵ_{rel}. The search is, thus terminated if either

$$f(\bar{\mathbf{x}}, \bar{\mathbf{y}}) - LB \leqslant \epsilon \quad \text{or} \quad \frac{f(\bar{\mathbf{x}}, \bar{\mathbf{y}}) - LB}{|f(\bar{\mathbf{x}}, \bar{\mathbf{y}})| + 10^{-10}} \leqslant \epsilon_{rel}$$

are satisfied. All tests ran on an Intel® Xeon® CPU (24 cores) 2.67 GHz server with 128GB of RAM running Ubuntu. For the termination criteria, we set the tolerances $\epsilon = 10^{-5}$ and $\epsilon_{rel} = 10^{-3}$, and a time limit of $900s$. The multi-tree results are run with up to 8 threads, while the single-tree results are run with a single thread.

The LP/NLP-based B&B algorithm can be implemented through so-called call-backs to the MILP solver in the B&B process, both to solve the continuous nonlinear subproblems and add new cuts to the LP problems in the B&B search. The initialization procedure is the same as in traditional OA to set up the first problem OA-MILP. For each feasible integer solution $\hat{\mathbf{y}}$ found in the B&B process, it is checked whether that specific integer combination has been found earlier in the search, *i.e.,* $\hat{\mathbf{y}} \in \{\mathbf{y}^i | i = 1, \dots, k-1\}$. If $\hat{\mathbf{y}} \notin \{\mathbf{y}^i | i = 1, \dots, k-1\}$, then the $\mathbf{x}^k$ variables can be obtained by solving NLP-I using that integer combination. If NLP-I is infeasible, the feasibility problem NLP-f is solved to determine the continuous variables. Cuts are generated according to (1) and added to all open nodes in the search tree, and practically implemented as lazy constraints. If $\hat{\mathbf{y}} \in \{\mathbf{y}^i | i = 1, \dots, k-1\}$, then there is no need to again solve NLP-I as the cuts already added for this integer combination is sufficient for the linear approximation to be tight for this integer combination [6]. This situation can, for example, occur if two different feasible solutions to the original problem only differ in the values of the continuous variables. This situation can be avoided by adding no-good cuts at every found solution [4].

7.2 Detailed examples

We first present detailed results for six particular instances of the selected test set. These problems were chosen to illustrate in detail the advantage of the ROA methods. The selected instances are `cvxnonsep_normcon20`[3], `cvxnonsep_psig40`[4], `nvs11`[5], `nvs12`[6], `slay08m`[7], and `smallinvDAXr1b150-165`[8] .

The statistics of these instances and their solution details are presented in Table 1. `slay08m` corresponds to the Big-M formulation of a safety layout problem, introduced in [47]. This instance is a Mixed-binary Quadratically Constrained Program (MBQP). `cvxnonsep_normcon20` and `cvxnonsep_psig40` are numerical instances proposed by Kronqvist et al [33]. The first one considers a single norm-2 constraint of 10 integer and 10 continuous variables. The second one minimizes a signomial function in terms of integer and continuous variables, making them a Mixed-integer Quadratically Constrained Program (MIQCP) and a general MINLP, respectively. The `cvxnonsep` instances are designed to be particularly difficult for OA-type methods. `nvs11` and `nvs12` instances proposed by Gupta and Ravindran [21] that have been widely used for benchmarking MINLP solver, see *e.g.,* [50]. They contain only integer variables, and quadratic constraints, and

[3] `http://minlplib.org/cvxnonsep_normcon20.html`
[4] `http://www.minlplib.org/cvxnonsep_psig40.html`
[5] `http://www.minlplib.org/nvs11.html`
[6] `http://www.minlplib.org/nvs12.html`
[7] `http://minlplib.org/slay08m.html`
[8] `http://minlplib.org/smallinvDAXr1b150-165.html`

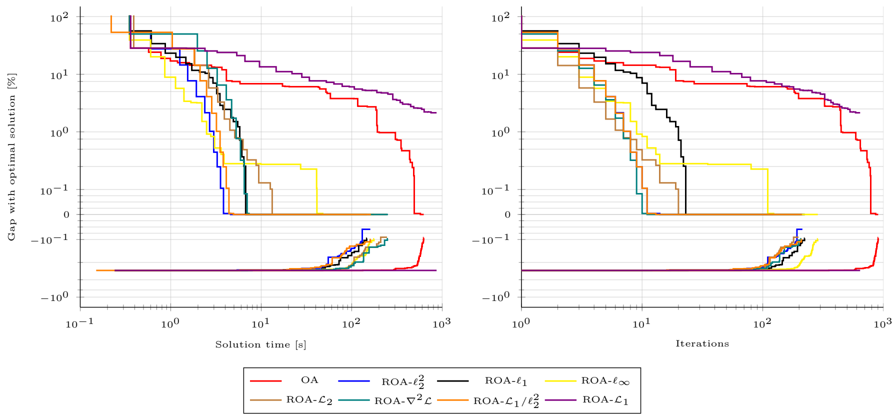

Fig. 5: Bound profiles for instance `cvxnonsep_psig40` against (a) solution time and (b) iterations using the multi-tree ROA method as described in Algorithm 1. The figure shows the upper and lower bounds obtained by the different regularization methods.

objective function; making them Integer Quadratically Constrained Quadratic Programs (IQCQP). `smallinvDAXr1b150-165` models an Extension of the Markovitz Mean-Variance-Optimization model by constraints for small investors. These problems belong to MIQCP.

To illustrate how the methods differ for these problems, we first show the upper and lower bounds obtained by each method in Figures 5 and 6. Each figure shows the percentage gap with the known optimal solution with respect to time and iterations. These plots have a semi-log vertical axis, where the values within $[-\epsilon_{rel}, \epsilon_{rel}]$ are presented in a linear scale, while values beyond that are presented in a logarithmic scale.

Figure 5 shows the progress of the bounds as a function of time and iterations for problem `cvxnonsep_psig40` in the multi-tree setting. We observe that the UB is quickly reduced to the optimal solution by the regularization methods compared to OA, except for ROA-$\mathcal{L}_1$, corresponding to the previous observation that the gradient of the Lagrangean does not provide a stabilization center, hence performing worse than the other methods. This was the only approach unable to converge within the time limit in the multi-tree setting. The LB is then improved to reach convergence within the specified optimality tolerances. When observing the bounds progress with respect to the iterations, the difference is even more drastic, showing the positive effect of regularization on this problem.

The bounds profiles for all the presented methods through a single-tree implementation when solving problem `cvxnonsep_normcon20` are presented in Figure 6. Contrary to problem `cvxnonsep_psig40`, the optimal solution is found by all methods in the first iteration, leaving the remaining task to improve the LB until the gap is within the specified tolerance. Although the

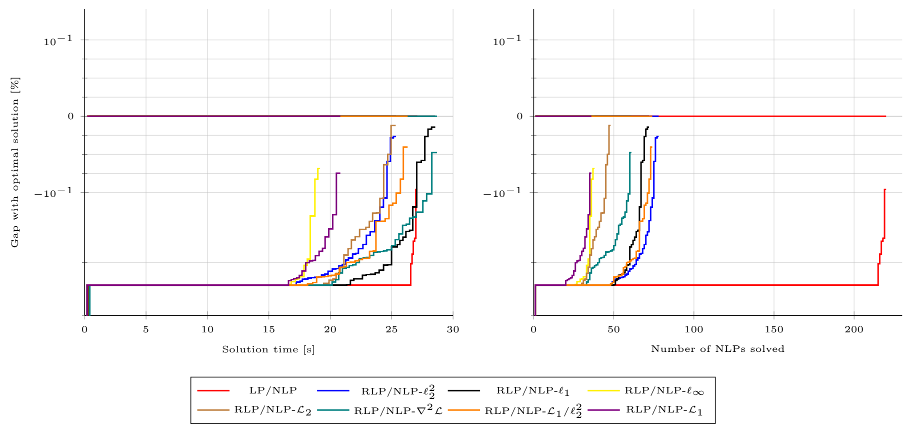

Fig. 6: Bound profiles for instance `cvxnonsep_normcon20` against (a) solution time and (b) NLP problems solved using the single-tree RLP/NLP methods as described in Algorithm 2. The figure shows the upper and lower bounds obtained by the different regularization methods.

regularization problems address the UB improvement part of OA directly by providing integer combinations close to stabilization centers defined by the incumbent solutions, we see that they also favor the more efficient convergence of the LB. This effect is best observed in the bounds plot against the number of NLPs solved, considered for the single-tree as a measure of iterations. The regularization methods require only a fraction of the NLP subproblems to obtain a LB within the optimality tolerance. This difference is not as prominent in terms of computational time. However, for this problem, the most efficient regularization method ROA-ℓ_∞ reduces the run time by approximately a third.

Table 1 presents a more detailed view of the results for the different examples. Here we notice that, although the ROA method spends extra time to solve the regularization problem, this eventually reduces the total solution time compared to OA. Instance `cvxnonsep_normcon20` shows a positive effect of the regularization methods, where the number of infeasible NLP problems is drastically reduced from 175 and 20 in the multi-tree and single-tree cases, respectively, to zero in all regularization cases. This leads to an advantage of the regularization methods against OA for this instance. A similar situation happens with instances `nvs11` and `nvs12`, where all regularization methods reduce the number of infeasible NLPs compared to OA, with the exception of ROA-$\mathcal{L}_1$. This supports the notion that the gradient of the Lagrangean does not define a stabilization center in the regularization objective; therefore, it is not a regularization per se. Moreover, using the gradient of the Lagrangean as a regularization objective (16) fails to converge to the optimal solution of examples `slay08m` and `smallinvDAXr1b150-165` in the single-tree implementation and of `cvxnonsep_psig40` in the both single- and multi-tree implementations. However, OA converged in a little under 8 minutes. This highlights the advantages of flexible implementation that allow these different approaches to be simply activated or deactivated.

As a general observation, the regularization methods reduce the respective number of iterations (number of NLP subproblems) required for convergence compared to the case without a regularization. The time spent solving those mixed-integer subproblems reduces the total algorithm time in most cases. These methods also tend to find the optimal solutions more quickly, having a practical effect if the time limit exceeds the time required to solve the problems. Besides, finding reasonable feasible solutions early in the search leads to tighter linearizations to the polyhedral approximation of the continuously relaxed nonlinear feasible region, leading to a better LB and faster convergence of the methods.

7.3 Numerical results

Based on the ROA method's good performance in the previous section, we perform a benchmark on Problem Set 1 and Problem Set 2. The software Paver 2 [10] is used to analyze the performance of the different methods proposed in this work. We decide to present our results in the form of absolute performance profiles, as seen in Figures 7, 8, 9 and 10. These plots show the total number of instances found to be solved within 0.1% of the known optimal solution of the problem against a measure of algorithmic effort, either solution time or iterations. These figures include two extra lines, where the "Virtual best" and "Virtual worst" alternatives are included. These cases are constructed with the best and worst solvers for each instance. Notice that we define iterations in the single-tree context as the number of NLP-I problems solved.

Figures 7 and 8 show the performance of the multi-tree implementation of the different methods for the highly nonlinear instances, defined according to (17). In general, the regularization methods achieve a better performance in terms of solution time and iterations than OA. For simple examples, highlighted on the left side of the profiles and given by instances solvable in less than 10 seconds or requiring fewer than ten iterations, OA seems to outperform most of the regularization methods, except for ROA-$\mathcal{L}_2$. This method, called Q-OA in [36] performs almost as the Virtual best solver in terms of iterations, demonstrating the value of incorporating the constraint curvature information in the regularization via the second-order Taylor approximation of the Lagrangean. In terms of solution time, the advantages of this approach are reduced given the complexity associated with obtaining the Hessian of the Lagrangean and, more importantly, addressing an MIQP regularization problem. Toward the end of the time limit, the other regularization methods catch up to the performance of ROA-$\mathcal{L}_2$, with ROA-$\nabla^2\mathcal{L}$ being able to solve 104 problems, the most among all the methods, after 15 minutes. Note that the worst method is ROA-$\mathcal{L}_1$, which, as mentioned above, is not an actual regularization method given that its projection objective function does not induce a

Table 1: Itemized results of the detailed examples.

Instance (# of discrete vars., # of continuous vars., # of const.)	Regularization method	ROA								RLP/NLP						
		Iterations	Time [s]	MILP time [s]	Regularization time [s]	NLP time [s]	Best solution found time [s]	Infeasible NLPs	Relative opt. gap	Nodes	Time [s]	Regularization time [s]	NLP time [s]	Best solution found time [s]	Infeasible NLPs	Relative opt. gap
cvxnonsep_normcon20 (10, 10, 1)	None	426	148.9	67.7	-	70.5	0.2	175	0.09%	239	27.2	-	19.8	24.5	20	0.01%
	ℓ_2^2	**64**	**19.3**	**5.6**	**7**	**5.3**	**0.2**	**0**	**0.07%**	77	25.6	6.6	5.9	0.2	0	0.01%
	ℓ_1	65	22.2	5.9	6.8	7.4	0.3	0	0.10%	71	28.7	10.5	5.3	0.4	0	0.01%
	ℓ_∞	103	35.8	10.5	10.6	11.5	0.6	0	0.07%	**37**	**19.4**	**1.6**	**2.7**	**0.2**	**0**	**0.01%**
	$\mathcal{L}_2$	67	31.7	6.2	10.9	8.7	0.4	0	0.08%	47	25.6	4.2	4.4	0.2	0	0.01%
	$\nabla^2\mathcal{L}$	64	27.4	5.3	9.4	7.4	0.3	0	0.07%	60	29	5.6	5.3	0.4	0	0.01%
	$\mathcal{L}_1/\ell_2^2$	62	20.9	3.2	6.8	7	0.4	0	0.10%	73	26.5	6.2	6.7	0.2	0	0.01%
	$\mathcal{L}_1$	125	43.4	13.1	16.3	10.4	0.3	0	0.10%	35	21.1	2.2	3	0.2	0	0.01%
cvxnonsep_psig40 (20, 20, 0)	None	905	667.5	512.5	-	112.8	612.9	0	0.09%	**857**	**462.1**	**-**	**191.2**	**274.1**	**0**	**0.01%**
	ℓ_2^2	213	124.3	35	66.8	15.8	4.8	0	0.06%	173	899.5	112.4	30	194.5	0	0.34%
	ℓ_1	223	161.7	51.8	70.2	26	8.6	0	0.10%	166	899.7	107.9	29.8	204.7	0	0.34%
	ℓ_∞	286	172.2	66.4	59	31.8	61.3	0	0.10%	202	900.5	52.5	61	26.3	0	0.34%
	$\mathcal{L}_2$	**195**	**306**	**54.2**	**180.9**	**29.3**	**22.3**	**0**	**0.07%**	3	899.4	0.8	1.2	33	0	0.34%
	$\nabla^2\mathcal{L}$	209	203.3	38	113.3	22.7	6.8	0	0.10%	3	899.3	0.7	0.9	26.4	0	0.34%
	$\mathcal{L}_1/\ell_2^2$	211	195.1	21.2	103	29.3	7.9	0	0.10%	2	899.4	0.3	0.9	0.7	0	0.35%
	$\mathcal{L}_1$	773	901.3	241.6	542	73.6	877.2	0	1.78%	2	899.4	0.3	0.9	0.7	0	0.35%
nvs11 (3, 0, 3)	None	24	5.1	1.2	-	3.2	4.2	13	0.00%	18	5.5	-	4.9	0.6	12	0.00%
	ℓ_2^2	13	2.9	0.6	0.6	1.5	2.3	3	0.00%	6	2.2	0.5	1.4	0.4	3	0.00%
	ℓ_1	**14**	**2.8**	**0.6**	**0.5**	**1.3**	**1.8**	**3**	**0.00%**	6	2.2	0.5	1.4	0.4	3	0.00%
	ℓ_∞	13	3.4	0.6	0.8	1.6	2.6	3	0.00%	6	3.2	0.6	2.1	0.6	2	0.00%
	$\mathcal{L}_2$	9	2.9	0.5	0.7	1.1	1	3	0.00%	5	2.8	0.5	1.6	0.6	2	0.00%
	$\nabla^2\mathcal{L}$	14	3.7	0.6	1.1	1.4	2.4	3	0.00%	5	2.4	0.5	1.4	0.5	2	0.00%
	$\mathcal{L}_1/\ell_2^2$	13	5.6	0.3	1.5	2.7	4	3	0.00%	5	3.2	0.7	2	0.7	2	0.00%
	$\mathcal{L}_1$	27	7.5	1.2	2.3	3.1	4.9	18	0.00%	**2**	**1.7**	**0.3**	**1**	**0.7**	**2**	**0.00%**
nvs12 (4, 0, 4)	None	26	6.2	1.6	-	3.6	5.9	13	0.00%	23	7.1	-	6.3	0.5	13	0.00%
	ℓ_2^2	16	3.8	0.8	1	1.5	2.8	4	0.00%	6	2.2	0.4	1.3	0.4	3	0.00%
	ℓ_1	16	3.5	0.7	0.7	1.6	2.4	4	0.00%	8	2.9	0.6	1.8	0.5	4	0.00%
	ℓ_∞	14	3.5	0.8	0.9	1.3	2.6	3	0.00%	6	3.6	0.7	2.2	0.7	1	0.00%
	$\mathcal{L}_2$	**9**	**3.3**	**0.6**	**0.8**	**1.3**	**1.2**	**3**	**0.00%**	6	3.4	0.6	1.9	0.6	3	0.00%
	$\nabla^2\mathcal{L}$	16	4.6	1	1.2	1.6	3	3	0.00%	10	4.2	0.9	2.5	0.5	7	0.00%
	$\mathcal{L}_1/\ell_2^2$	14	6.2	0.8	1.6	2.6	4.5	3	0.00%	6	4	0.9	2.3	0.7	3	0.00%
	$\mathcal{L}_1$	33	10.1	2.2	2.7	3.8	9.4	20	0.00%	**2**	**1.8**	**0.3**	**1**	**0.7**	**2**	**0.00%**
slay08m (112, 72, 252)	None	24	24.4	10.7	-	8.2	20.4	0	0.07%	50	89.3	-	23.3	83.4	0	0.00%
	ℓ_2^2	**13**	**15**	**4.3**	**4.3**	**3.8**	**12.6**	**0**	**0.07%**	23	68.2	21.1	8.7	56.4	0	0.00%
	ℓ_1	16	16.5	4.9	3.8	4.3	7.2	0	0.07%	20	58.5	7.9	8.1	53.4	0	0.01%
	ℓ_∞	17	20.9	6	5.6	5.5	6.3	0	0.00%	15	88.5	7.1	7.9	77.5	0	0.00%
	$\mathcal{L}_2$	18	51.1	8.1	6.3	6.9	16.3	0	0.07%	22	92.4	7.4	8.9	24.5	0	0.00%
	$\nabla^2\mathcal{L}$	12	21.3	2.8	2.4	2.8	19.7	0	0.07%	22	100	8.2	7.7	93.9	0	0.00%
	$\mathcal{L}_1/\ell_2^2$	13	24	5.7	6.9	4.9	7.3	0	0.07%	**18**	**57.2**	**9.2**	**8.7**	**54.6**	**0**	**0.01%**
	$\mathcal{L}_1$	55	94.4	24.6	28.8	21.1	92.8	0	0.00%	23	83.1	5.7	11.1	51.7	0	31.65%
smallinvDAXr1b150-165 (30, 1, 4)	None	98	51.3	18.1	-	24.9	51.3	0	0.10%	119	64.4	-	52.6	62	0	0.01%
	ℓ_2^2	14	9.2	1.7	2.3	3.3	9.2	0	0.09%	**27**	**22.3**	**5.7**	**9.5**	**15.7**	**0**	**0.01%**
	ℓ_1	16	13.6	2.8	2.6	5.1	13.6	0	0.05%	36	28.3	7.7	13.5	11.4	0	0.01%
	ℓ_∞	18	15.6	3.2	3.3	6	15.5	0	0.07%	36	34.3	9.3	17.4	15	0	0.01%
	$\mathcal{L}_2$	**2**	**2.3**	**0.3**	**0.4**	**0.6**	**2.3**	**0**	**0.02%**	20	30.1	16.3	7.6	3.5	0	0.01%
	$\nabla^2\mathcal{L}$	13	9.9	1.3	2.7	3	9.9	0	0.06%	26	23.8	8.1	7.9	13.3	0	0.01%
	$\mathcal{L}_1/\ell_2^2$	14	10.8	0.6	2.8	4.1	10.8	0	0.08%	29	30	8.2	13	17	0	0.01%
	$\mathcal{L}_1$	225	265.1	57.6	82.2	79.4	265.1	0	0.09%	15	16.8	2.6	7	16.2	0	16.28%

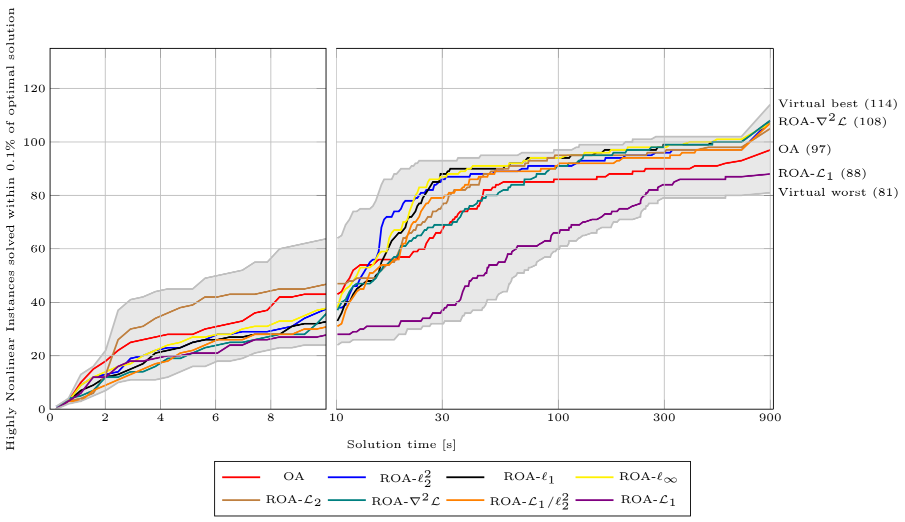

Fig. 7: Time performance profile for highly nonlinear instances for multi-tree ROA method as described in Algorithm 1.

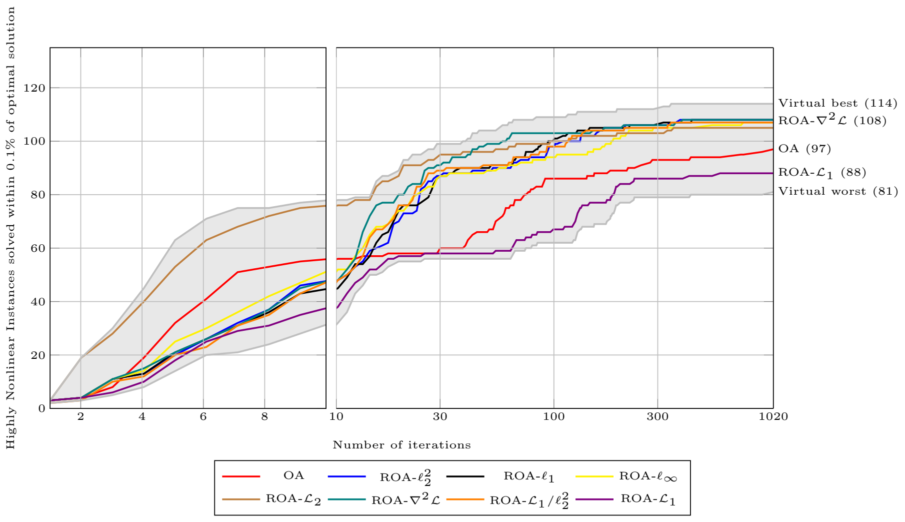

Fig. 8: Iteration performance profile for highly nonlinear instances for multi-tree ROA method as described in Algorithm 1.

stabilization center. Traditional OA can solve 97 of the 135 problems to 0.1% of the optimal solution in Problem Set 2 within 900 seconds.

When considering all convex MINLP in MINLPLib, Problem Set 1, the gap between regularization-based methods and OA reduces, mainly since most of these instances have low nonlinearity. Some alternatives of regularization methods solve more instances than OA, with ROA-ℓ_1 solving 330 out of the 438 instances within the time limit, 10 more than OA. The performance profiles for the instances in Problem Set 1 are included in the Appendix in Figures 11 and 12. Besides the improved performance of the ROA methods compared to OA, we could not clearly identify any features in the problems that would benefit one regularization norm compared to the others.

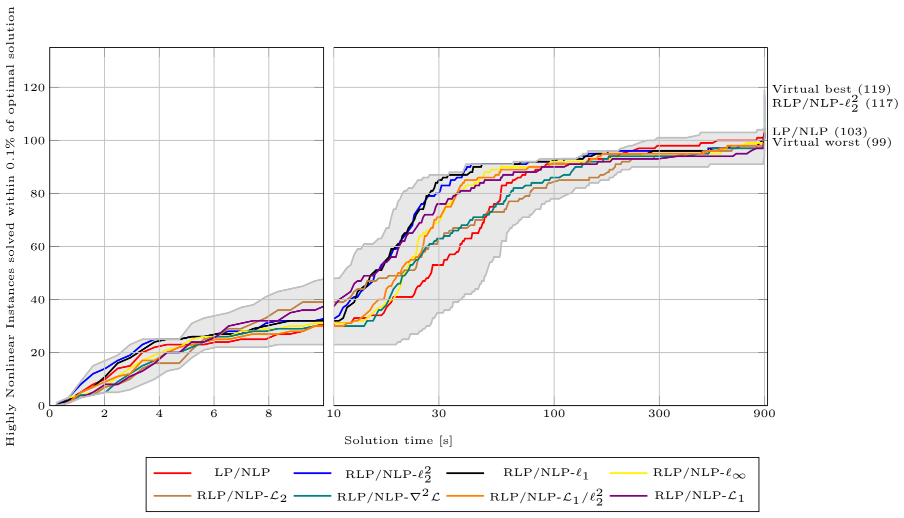

Fig. 9: Time performance profile for highly nonlinear instances for single-tree RLP/NLP methods as described in Algorithm 2.

The performance profiles for the Problem Set 2 of RLP/NLP are shown in Figures 9 and 10. In terms of NLP subproblems solved, the OA method is almost equivalent to the virtual worst, demonstrating that the regularization approaches lead to a more meaningful solution of NLP problems in the solution procedure. This observation is not directly translated into the time profiles, considering that solving an extra mixed-integer program for every incumbent solution in the tree is an expensive step, although justifiable with reducing iterations. Considering Problem set 2, the most successful approach is RLP/NLP-ℓ_2^2 being able to solve 117 instances, 15 more than OA.

When considering Problem Set 1, the single-tree implementation of RLP/NLP-$\mathcal{L}_2$ solves the least number of instances within the time limit. This contrasts with the good performance this regularization had for the multi-tree implementation. In multi-tree and single-tree, when considering all the convex MINLP instances from MINLPLib, the most successful regularization type was ℓ_1. This demonstrates the potential that linearly representable regularizations have in terms of performance.

Out of Problem Set 2, 114 problems out of 135 could be solved by all methods, and 21 could not be solved using the multi-tree methods. All single-tree

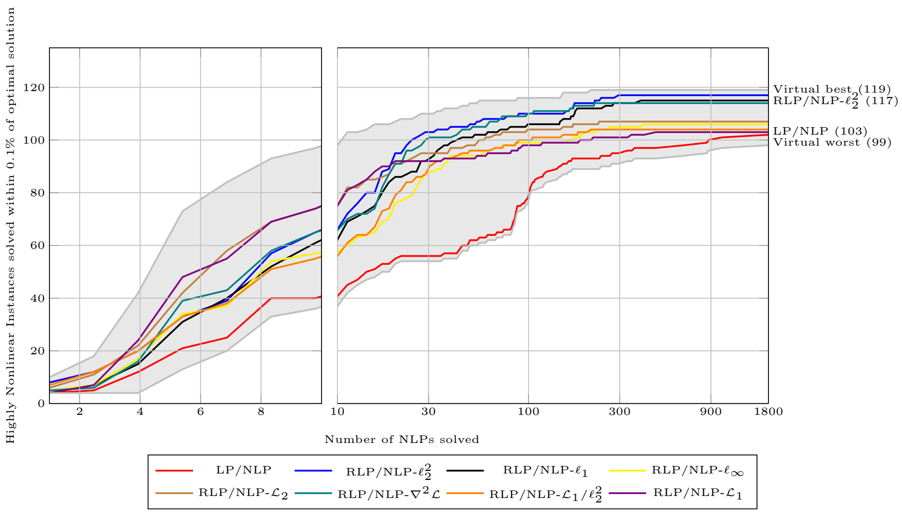

Fig. 10: Iteration performance profile for highly nonlinear instances for single-tree RLP/NLP methods as described in Algorithm 2.

Table 2: Details for each method infeasible subproblems when solving Problem Set 1 (438 instances).

	ROA		RLP/NLP			
Regularization method	# of instances with infeasible NLP-I	Fraction of infeasible NLP-I	# of instances with infeasible NLP-I	Fraction of infeasible NLP-I	# of instances with infeasible MIP-Proj	Fraction of infeasible MIP-Proj
None	111	3743/15671=23.9%	122	4211/26832=15.7%	-	-
ℓ_2^2	104	2395/8278=28.9%	110	2067/7544=27.4%	290	2540/6765=37.5%
ℓ_1	104	2670/9031=29.6%	114	1980/8384=23.6%	348	3084/8203=37.6%
ℓ_∞	101	2671/10517=25.4%	109	2469/8559=28.8%	344	2752/7989=34.4%
$\mathcal{L}_2$	103	2359/6749=35.0%	112	1558/5974=26.1%	283	2157/5161=41.8%
$\nabla^2\mathcal{L}$	104	2435/7101=34.3%	110	1539/6171=24.9%	284	2270/5437=41.8%
$\mathcal{L}_1/\ell_2^2$	100	2210/7731=28.6%	112	1501/6400=23.5%	278	2140/5556=38.5%
$\mathcal{L}_1$	110	2878/17041=16.9%	118	1459/7176=20.3%	262	2429/6311=38.5%

methods solved 119 of these instances, and none could solve 16 of those instances. For the whole test set, Problem Set 1, 342 instances of 438 were solved by all methods, and 96 were not solved by any in the multi-tree implementation. The single-tree methods were slightly more successful, with 351 cases solved by all and 87 by none.

As similarly found in [36], the number of infeasible NLP-I problems encountered diminished when using regularization methods. Both in the multi- and single-tree implementations, the regularization approaches were able to solve fewer instances requiring the solution of problem NLP-f compared to OA.

An exciting finding of Table 2 is that the fraction of problems NLP-I that were infeasible was larger for the regularization methods! This result was surprising given the hypothesis that is choosing an integer combination close to a feasible solution results in trial solutions close to the feasible region, resulting in fewer infeasible trial solutions. The explanation for this behavior has to do with the total number of NLP-I problems solved by each method. Being the OA iterations less time-consuming more could be performed within the time limit. This would set the denominator in the fraction to be large, making the fraction smaller than for the other methods. This is not a measure to be considered independently from the previous results. The fewer iterations a problem requires to converge, the fewer NLP-I problems it needs to solve. Therefore, many solved NLPs indicate inefficient methods, although it would decrease the fraction of infeasible subproblems encountered. This can be observed with ROA-$\mathcal{L}_1$, whose fraction of infeasible NLPs is the lowest among all the tested methods. However, it was the weakest alternative considered in this manuscript.

For the single-tree implementation, we observed that Remark 2 often appeared in practice, with three-quarters of the instances presenting at some point infeasible MIP-Proj problems. These infeasible problems arise from the weak LB coming from the B&B tree, leading to an average of 40% of all MIP-Proj problems being infeasible in the single-tree setting.

8 Conclusions and future work

This manuscript presents a new solution framework for multi-tree and single-tree Outer-Approximation based on regularizations for solving convex MINLP problems. We present seven different regularization methods for OA through this framework, including two that were presented earlier in [36]. These regularizations can be classified into two groups: those based on distance minimization around an incumbent solution and those based on approximations of the Lagrangean function around that incumbent solution. The regularization approach relies on the solution of an auxiliary mixed-integer program. Based on the objective function's choice, it can be a mixed-integer linear program or a mixed-integer quadratic program. We show that the convergence proofs from [36] directly apply to these methods as well, thus, guaranteeing convergence to the optimal solution. Moreover, the regularization ideas are integrated with the LP/NLP Branch & Bound method [46] leading to a single-tree regularization algorithm for convex MINLP. The implementation of these methods was done on top of the Mixed-Integer Decomposition Toolbox for Pyomo - MindtPy [3] in open-source code. We evaluated these approaches experimentally and compared them to OA by solving a large set of convex MINLP problems. We observed that the regularization approaches are especially well-fitted for highly nonlinear problems, achieving performance improvements compared to OA. This confirms the hypothesis that staying close to the feasible solutions ensures the integer combinations found by the linearizations to stay close to the convex set defined by the nonlinear constraints. For almost linear instances, the benefits of the regularization technique are sometimes lost due to the cost of solving auxiliary projection problems, which also aligns well with the results in [36]. However, our results demonstrate that using linearly representable regularizations does improve the average performance for all the convex MINLP instances at the benchmarking library MINLPLib, including the highly linear ones.

For future work, we consider an interesting avenue to perform updates in the Hessian of the Lagrangean estimate. As in the BFGS algorithm [17], the Hessian of the Lagrangean needs not to be computed exactly, and its approximation can be iteratively refined with the first estimate of it being a scaled identity matrix. This technique has proved extremely useful in trust-region methods for continuous NLP problems, such as Sequential Quadratic Programming (SQP) [12, 22]. Although our results do not show a clear advantage of certain regularization norms depending on problem features, we presented a flexible implementation for the different norms such that one can easily experiment with the different alternatives presented herein. By not evaluating the problems in the library MINLPLib, but randomly generating problems with controlled features, one can obtain experimental results to highlight which regularization norm is more efficient for a particular problem class. Moreover, the two extra parameters introduced in the regularization methods, namely α and d, have been maintained constant throughout these experiments. These hyperparameters represent a trade-off between how trustworthy the incumbent solution is compared to the optimal solution and how much exploration far from that incumbent solution is required. One can imagine a dynamic update policy for these parameters, balancing the incumbent solutions' exploration and exploitation as a future research direction. [9]

Acknowledgements David E. Bernal and Ignacio E. Grossmann would like to thank the Center Advanced Process Decision Making (CAPD) for its financial support. David E. Bernal also acknowledges USRA NASA Academic Mission Services (contract NNA16BD14C). Zedong Peng is grateful for the financial support from the China Scholarship Council (CSC) (No. 201906320320). Jan Kronqvist is grateful for the Newton International Fellowship by the Royal Society (NIF\R1\182194), a grant by the Swedish Cultural Foundation in Finland, and support by Digital Futures at KTH.

References

1. Abhishek K, Leyffer S, Linderoth J (2010) FilMINT: An outer approximation-based solver for convex mixed-integer nonlinear programs. INFORMS Journal on computing 22(4):555–567
2. Bagirov A, Karmitsa N, Mäkelä MM (2014) Introduction to Nonsmooth Optimization: Theory, Practice and Software. Springer
3. Bernal DE, Chen Q, Gong F, Grossmann IE (2018) Mixed-integer nonlinear decomposition toolbox for Pyomo (MindtPy). In: Computer Aided Chemical Engineering, vol 44, Elsevier, pp 895–900
4. Bernal DE, Vigerske S, Trespalacios F, Grossmann IE (2020) Improving the performance of DICOPT in convex MINLP problems using a feasibility pump. Optimization Methods and Software 35(1):171–190

[9] The datasets generated during and analysed during the current study are available in the GitHub repository, https://zedongpeng.github.io/ROA-RLPNLP-Benchmark/

5. Boggs PT, Tolle JW (1995) Sequential quadratic programming. Acta numerica 4:1–51
6. Bonami P, Biegler LT, Conn AR, Cornuéjols G, Grossmann IE, Laird CD, Lee J, Lodi A, Margot F, Sawaya N, Wächter A (2008) An algorithmic framework for convex mixed integer nonlinear programs. Discrete Optimization 5(2):186–204, DOI 10.1016/j.disopt.2006.10.011
7. Bonami P, Cornuéjols G, Lodi A, Margot F (2009) A feasibility pump for mixed integer nonlinear programs. Mathematical Programming 119(2):331–352
8. Boukouvala F, Misener R, Floudas CA (2016) Global optimization advances in mixed-integer nonlinear programming, MINLP, and constrained derivative-free optimization, CDFO. European Journal of Operational Research 252(3):701–727
9. Bussieck MR, Drud AS, Meeraus A (2003) MINLPLib—a collection of test models for mixed-integer nonlinear programming. INFORMS Journal on Computing 15(1):114–119
10. Bussieck MR, Dirkse SP, Vigerske S (2014) PAVER 2.0: an open source environment for automated performance analysis of benchmarking data. Journal of Global Optimization 59(2-3):259–275
11. Coey C, Lubin M, Vielma JP (2020) Outer approximation with conic certificates for mixed-integer convex problems. Mathematical Programming Computation pp 1–45
12. Conn AR, Gould NI, Toint PL (2000) Trust region methods. SIAM
13. Dakin RJ (1965) A tree-search algorithm for mixed integer programming problems. The Computer Journal 8(3):250–255
14. De Mauri M, Gillis J, Swevers J, Pipeleers G (2020) A proximal-point outer approximation algorithm. Computational Optimization and Applications 77(3):755–777
15. Delfino A, de Oliveira W (2018) Outer-approximation algorithms for nonsmooth convex MINLP problems. Optimization 67(6):797–819
16. Duran MA, Grossmann IE (1986) An outer-approximation algorithm for a class of mixed-integer nonlinear programs. Mathematical Programming 36(3):307–339
17. Fletcher R (2013) Practical Methods of Optimization. John Wiley & Sons
18. Fletcher R, Leyffer S (1994) Solving mixed integer nonlinear programs by outer approximation. Mathematical Programming 66(1):327–349
19. Floudas CA (1995) Nonlinear and mixed-integer optimization: fundamentals and applications. Oxford University Press
20. Geoffrion AM (1972) Generalized Benders decomposition. Journal of Optimization Theory and Applications 10(4):237–260
21. Gupta OK, Ravindran A (1985) Branch and bound experiments in convex nonlinear integer programming. Management science 31(12):1533–1546
22. Han SP (1976) Superlinearly convergent variable metric algorithms for general nonlinear programming problems. Mathematical Programming 11(1):263–282

23. Hart WE, Laird CD, Watson JP, Woodruff DL, Hackebeil GA, Nicholson BL, Siirola JD, et al (2017) Pyomo-optimization modeling in python, vol 67. Springer
24. den Hertog D, Kaliski J, Roos C, Terlaky T (1995) A logarithmic barrier cutting plane method for convex programming. Annals of Operations Research 58(2):67–98
25. HSL (2007) A collection of Fortran codes for large scale scientific computation. URL http://www.hsl.rl.ac.uk
26. Hunting M (2011) The AIMMS outer approximation algorithm for MINLP. Tech. rep., AIMMS B.V.
27. IBM Corp, IBM (2020) V20.1: User's Manual for CPLEX. International Business Machines Corporation URL https://www.ibm.com/docs/en/icos/20.1.0?topic=cplex
28. Kelley JE Jr (1960) The cutting-plane method for solving convex programs. Journal of the Society for Industrial & Applied Mathematics 8(4):703–712
29. Khajavirad A, Sahinidis NV (2018) A hybrid LP/NLP paradigm for global optimization relaxations. Mathematical Programming Computation 10(3):383–421
30. Kiwiel KC (1995) Proximal level bundle methods for convex nondifferentiable optimization, saddle-point problems and variational inequalities. Mathematical Programming 69(1-3):89–109
31. Kreyszig E (1978) Introductory functional analysis with applications, vol 1. wiley New York
32. Kronqvist J, Lundell A, Westerlund T (2016) The extended supporting hyperplane algorithm for convex mixed-integer nonlinear programming. Journal of Global Optimization 64(2):249–272
33. Kronqvist J, Lundell A, Westerlund T (2018) Reformulations for utilizing separability when solving convex minlp problems. Journal of Global Optimization 71(3):571–592
34. Kronqvist J, Bernal DE, Lundell A, Grossmann IE (2019) A review and comparison of solvers for convex MINLP. Optimization and Engineering 20(2):397–455
35. Kronqvist J, Bernal DE, Lundell A, Westerlund T (2019) A center-cut algorithm for quickly obtaining feasible solutions and solving convex MINLP problems. Computers & Chemical Engineering 122:105–113
36. Kronqvist J, Bernal DE, Grossmann IE (2020) Using regularization and second order information in outer approximation for convex MINLP. Mathematical Programming 180(1):285–310
37. Lee J, Leyffer S (2011) Mixed integer nonlinear programming, vol 154. Springer Science & Business Media
38. Lemaréchal C, Nemirovskii A, Nesterov Y (1995) New variants of bundle methods. Mathematical Programming 69(1-3):111–147
39. Liberti L, Marinelli F (2014) Mathematical programming: Turing completeness and applications to software analysis. Journal of Combinatorial Optimization 28(1):82–104

40. Liberti L, Cafieri S, Tarissan F (2009) Reformulations in mathematical programming: A computational approach. In: Foundations of Computational Intelligence Volume 3, Springer, pp 153–234
41. Lundell A, Kronqvist J (2019) Integration of polyhedral outer approximation algorithms with mip solvers through callbacks and lazy constraints. In: AIP Conference Proceedings, AIP Publishing LLC, vol 2070, p 020012
42. Lundell A, Kronqvist J, Westerlund T (2022) The supporting hyperplane optimization toolkit for convex minlp. Journal of Global Optimization pp 1–41
43. Muts P, Nowak I, Hendrix EM (2020) The decomposition-based outer approximation algorithm for convex mixed-integer nonlinear programming. Journal of Global Optimization pp 1–22
44. Nesterov Y (2004) Introductory Lectures on Convex Optimization: A Basic Course, vol 87. Springer
45. de Oliveira W (2016) Regularized optimization methods for convex MINLP problems. TOP 24(3):665–692
46. Quesada I, Grossmann IE (1992) An LP/NLP based branch and bound algorithm for convex MINLP optimization problems. Computers & chemical engineering 16(10-11):937–947
47. Sawaya N, Grossmann IE (2008) Reformulations, relaxations and cutting planes for linear generalized disjunctive programming
48. Slater M (1950) Lagrange multipliers revisited. Tech. rep., Cowles Foundation for Research in Economics, Yale University
49. Su L, Tang L, Bernal DE, Grossmann IE (2018) Improved quadratic cuts for convex mixed-integer nonlinear programs. Computers & Chemical Engineering 109:77–95
50. Tawarmalani M, Sahinidis NV (2013) Convexification and global optimization in continuous and mixed-integer nonlinear programming: theory, algorithms, software, and applications, vol 65. Springer Science & Business Media
51. Trespalacios F, Grossmann IE (2014) Review of mixed-integer nonlinear and generalized disjunctive programming methods. Chemie Ingenieur Technik 86(7):991–1012
52. Wächter A, Biegler LT (2006) On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Mathematical Programming 106(1):25–57
53. Westerlund T, Petterson F (1995) An Extended Cutting Plane Method For Solving Convex MINLP Problems. Computers & Chemical Engineering 19:S131–S136

9 Appendix

9.1 Algorithmic description of OA and LP/NLP Branch & Bound

This section presents the algorithmic description of the Outer-Approximation method [16, 18], in Algorithm 3, and the LP/NLP Branch & Bound method [6, 46], in Algorithm 4.

Algorithm 3 An algorithm summarizing the Outer-Approximation method.

```
Define accepted optimality gap $\epsilon \geqslant 0$.
1. Initialization.
   1.1 Obtain a relaxed solution $\tilde{\mathbf{x}}, \tilde{\mathbf{y}}$ by solving an integer relaxation of the MINLP problem.
   1.2 Generate cuts at $\tilde{\mathbf{x}}, \tilde{\mathbf{y}}$ according to (1) and construct problem OA-MILP.
   1.3 Set iteration counter $k = 1$, $UB^0 = \infty$ and $LB^0 = -\infty$.
2. Repeat until $UB^{k-1} - LB^{k-1} \leqslant \epsilon$.
   2.1 Solve problem OA-MILP to obtain $\mathbf{y}^k$ and $LB^k$
   2.2 Solve problem NLP-I with integer variables fixed as $\mathbf{y}^k$ to obtain $\mathbf{x}^k$.
       2.2.1 If problem NLP-I is feasible, set $UB^k = \min\{f(\mathbf{x}^k, \mathbf{y}^k), UB^{k-1}\}$.
       2.2.2 If problem NLP-I is infeasible, obtain $\mathbf{x}^k$ by solving feasibility problem NLP-f
             and set $UB^k = UB^{k-1}$.
   2.3 Generate cuts at $\mathbf{x}^k, \mathbf{y}^k$ according to (1) and add these to problem OA-MILP.
   2.4 (Optional) Generate no-good cuts at $\mathbf{y}^k$ and add these to problems OA-MILP.
   2.5 Increase iteration counter, $k = k + 1$
3. Return the best found solution.
```

9.2 Representing ℓ_1 and ℓ_∞ norms using Linear Programming

This section shows the valid reformulations of optimization problems with norms one and infinity in the objective function using auxiliary variables and linear constraints. This reformulation is exact in the sense that they preserve the local and global optima from the original problem [40]. These reformulations are particularly interesting since they allow the regularization problem MIP-Proj to be written as Mixed-Integer Linear Programming (MILP) problems, instead of Mixed-Integer Quadratic Programming (MIQP) problems, as in the work byKronqvist et al [36]. MILP solution methods' maturity over MIQP allows these problems to be more quickly solvable in practice.

The norm-1 of a vector $\mathbf{v} \in V \subseteq \mathbb{R}^N$ whose components might be negative or positive, $\ell_1(\mathbf{v}) = \|\mathbf{v}\|_1 = \sum_{i=1}^N |v_i|$ can be reformulated in the case that this term appears in the objective function with a set of linear constraints. Through the addition of $2N$ non-negative slack variables $\mathbf{s}^+, \mathbf{s}^- \in \mathbb{R}_+^N$, and N linear equality constraints the following reformulation is valid:

Algorithm 4 An algorithm summarizing the LP/NLP based Branch & Bound algorithm.

```
Define accepted optimality gap $\epsilon \geqslant 0$.
 1. Initialization.
    1.1 Obtain a relaxed solution $\tilde{\mathbf{x}}, \tilde{\mathbf{y}}$ by solving an integer relaxation of the MINLP problem.
    1.2 Generate cuts at $\tilde{\mathbf{x}}, \tilde{\mathbf{y}}$ according to (1) and construct problems OA-MILP.
    1.3 Set node counter $k = 1$, $UB^0 = \infty$ and $LB^0 = -\infty$.
 2. Begin Branch & Bound for problem OA-MILP and terminate until $UB^{k-1} - LB^{k-1} \leqslant \epsilon$.
    2.1 If a new incumbent integer solution $\hat{\mathbf{x}}, \hat{\mathbf{y}}$ is found, check if $\hat{\mathbf{y}} \in \{\mathbf{y}^i | i = 1, \ldots, k-1\}$.
        2.1.1 if $\hat{\mathbf{y}} \in \{\mathbf{y}^i | i = 1, \ldots, k-1\}$, then skip this iteration and continue the Branch & Bound process.
        2.1.2 if $\hat{\mathbf{y}} \notin \{\mathbf{y}^i | i = 1, \ldots, k-1\}$, set $\mathbf{y}^k = \hat{\mathbf{y}}$ and set $LB^k$ to the lower bound of current B&B tree.
    2.2 Solve problem NLP-I with integer variables fixed as $\mathbf{y}^k$ to obtain $\mathbf{x}^k$.
        2.2.1 If problem NLP-I is feasible, set $UB^k = \min\{f(\mathbf{x}^k, \mathbf{y}^k), UB^{k-1}\}$.
        2.2.2 If problem NLP-I is infeasible, obtain $\mathbf{x}^k$ by solving feasibility problem NLP-f and set $UB^k = UB^{k-1}$.
    2.3 Generate cuts at $\mathbf{x}^k, \mathbf{y}^k$ according to (1) and add these as global lazy constraints to the B&B tree of problem OA-MILP.
    2.4 (Optional) Generate no-good cuts at $\mathbf{y}^k$ and add these as global lazy constraints to the B&B tree of problem OA-MILP.
    2.5 Increase node counter, $k = k + 1$.
 3 Return the best found solution.
```

$$
\begin{array}{ll} \min\limits_{\mathbf{v}} & \|\mathbf{v}\|_1 \\ \text{s.t.} & \mathbf{v} \in V \subseteq \mathbb{R}^N \end{array} \Leftrightarrow \begin{array}{ll} \min\limits_{\mathbf{v},\mathbf{s}^+,\mathbf{s}^-} & \sum_{i=1}^{N} s_i^+ + s_i^- \\ \text{s.t.} & \mathbf{s}^+ - \mathbf{s}^- = \mathbf{v} \\ & \mathbf{v} \in V \subseteq \mathbb{R}^N, \ \mathbf{s}^+ \in \mathbb{R}_+^N, \ \mathbf{s}^- \in \mathbb{R}_+^N \end{array} \tag{18}
$$

This reformulation is applied to the regularization problem MIP-Proj when considering the ℓ_1 regularization function as in (4), resulting in problem MIP-Proj-ℓ_1. It can also be potentially applied to the feasibility NLP problem NLP-f.

$$
\begin{aligned}
\min_{\mathbf{x},\mathbf{y},\mu,\mathbf{s}^+,\mathbf{s}^-} \quad & \sum_{j=1}^{n+m} s_j^+ + s_j^- \\
\text{s.t.} \quad & s_j^+ - s_j^- = x_j - \bar{x}_j \quad \forall j = \{1,\dots,n\} \\
& s_{n+j}^+ - s_{n+j}^- = y_j - \bar{y}_j \quad \forall j = \{1,\dots,m\} \\
& \mu \leqslant \hat{f}_k^\star \\
& f(\mathbf{x}^i,\mathbf{y}^i) + \nabla f(\mathbf{x}^i,\mathbf{y}^i)^\top \begin{bmatrix} \mathbf{x} - \mathbf{x}^i \\ \mathbf{y} - \mathbf{y}^i \end{bmatrix} \leqslant \mu \quad \forall i = 1,\dots,k, \\
& g_j(\mathbf{x}^i,\mathbf{y}^i) + \nabla g_j(\mathbf{x}^i,\mathbf{y}^i)^\top \begin{bmatrix} \mathbf{x} - \mathbf{x}^i \\ \mathbf{y} - \mathbf{y}^i \end{bmatrix} \leqslant 0 \quad \forall i = 1,\dots k, \forall j \in \mathcal{I}_i, \\
& \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} \leqslant \mathbf{b}, \\
& \mathbf{x} \in \mathbb{R}^n,\ \mathbf{y} \in \mathbb{Z}^m,\ \mu \in \mathbb{R},\ \mathbf{s}^+,\mathbf{s}^- \in \mathbb{R}_+^{n+m}
\end{aligned}
\tag{MIP-Proj-ℓ_1}
$$

The norm-∞ of a vector $\mathbf{v} \in V \subseteq \mathbb{R}^N$ whose components might be negative or positive, $\ell_\infty(\mathbf{v}) = \|\mathbf{v}\|_\infty = \max_{i=\{1,\dots,N\}} |v_i|$ can be reformulated in the case that this term appears in the objective function with a set of linear constraints. Through the addition of one non-negative slack variable $s \in \mathbb{R}_+$, and $2N$ linear inequality constraints, the following reformulation is valid:

$$
\begin{aligned}
\min_{\mathbf{v}} \quad & \|\mathbf{v}\|_\infty \\
\text{s.t.} \quad & \mathbf{v} \in V \subseteq \mathbb{R}^N
\end{aligned}
\Leftrightarrow
\begin{aligned}
\min_{\mathbf{v},s} \quad & s \\
\text{s.t.} \quad & s \geqslant \mathbf{v} \\
& s \geqslant -\mathbf{v} \\
& \mathbf{v} \in V \subseteq \mathbb{R}^N,\ s \in \mathbb{R}_+
\end{aligned}
\tag{19}
$$

This is the usual choice for reformulating problem NLP-f, and can also be used to reformulate problem MIP-Proj with ℓ_∞ regularization objective function, as in (5). This last problem formulation is:

$$
\begin{aligned}
\min_{\mathbf{x},\mathbf{y},\mu,s} \quad & s \\
\text{s.t.} \quad & s \geqslant x_j - \bar{x}_j \quad \forall j = \{1,\dots,n\} \\
& s \geqslant \bar{x}_j - x_j \quad \forall j = \{1,\dots,n\} \\
& s \geqslant y_j - \bar{y}_j \quad \forall j = \{1,\dots,m\} \\
& s \geqslant \bar{y}_j - y_j \quad \forall j = \{1,\dots,m\} \\
& \mu \leqslant \hat{f}_k^{\star} \\
& f(\mathbf{x}^i,\mathbf{y}^i) + \nabla f(\mathbf{x}^i,\mathbf{y}^i)^\top \begin{bmatrix} \mathbf{x}-\mathbf{x}^i \\ \mathbf{y}-\mathbf{y}^i \end{bmatrix} \leqslant \mu \quad \forall i = 1,\dots,k, \\
& g_j(\mathbf{x}^i,\mathbf{y}^i) + \nabla g_j(\mathbf{x}^i,\mathbf{y}^i)^\top \begin{bmatrix} \mathbf{x}-\mathbf{x}^i \\ \mathbf{y}-\mathbf{y}^i \end{bmatrix} \leqslant 0 \quad \forall i = 1,\dots k, \forall j \in \mathcal{I}_i, \\
& \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} \leqslant \mathbf{b}, \\
& \mathbf{x} \in \mathbb{R}^n,\ \mathbf{y} \in \mathbb{Z}^m,\ \mu \in \mathbb{R},\ s \in \mathbb{R}_+
\end{aligned}
\tag{MIP-Proj-ℓ_∞}
$$

9.3 Performance profiles for Problem Set 1

This section of the Appendix presents the performance profiles for the multi-tree and single-tree implementation of the methods included in this manuscript when solving all 358 convex MINLP problems in Problem Set 1. Figures 11 and 12 include the time and iteration performance profiles for the multi-tree implementation, respectively. Figures 13 and 14 include the time and iteration performance profiles for the single-tree implementation, respectively. Notice that we define iterations in the single-tree context as the number of NLP-I problems solved.

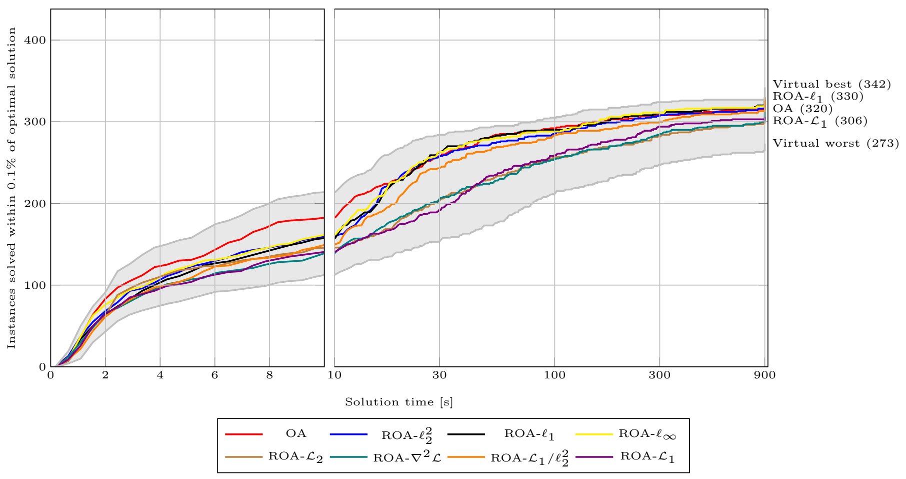

Fig. 11: Time performance profile for multi-tree ROA method as described in Algorithm 1.

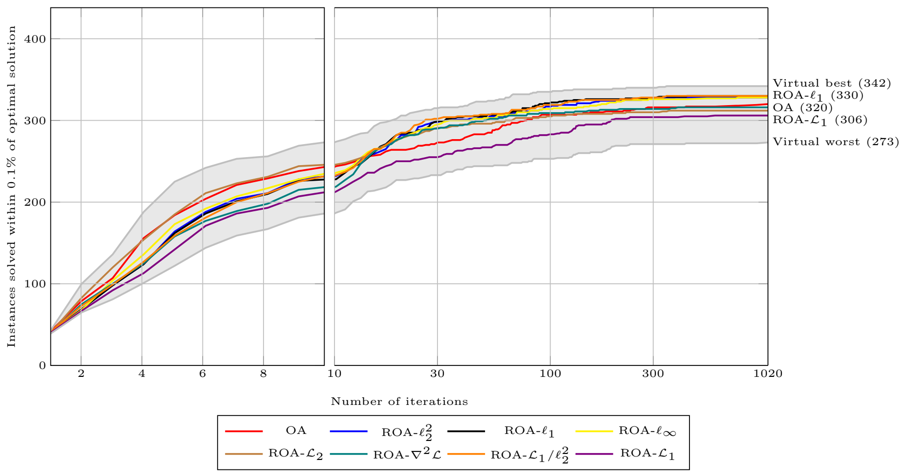

Fig. 12: Iteration performance profile for multi-tree ROA method as described in Algorithm 1.

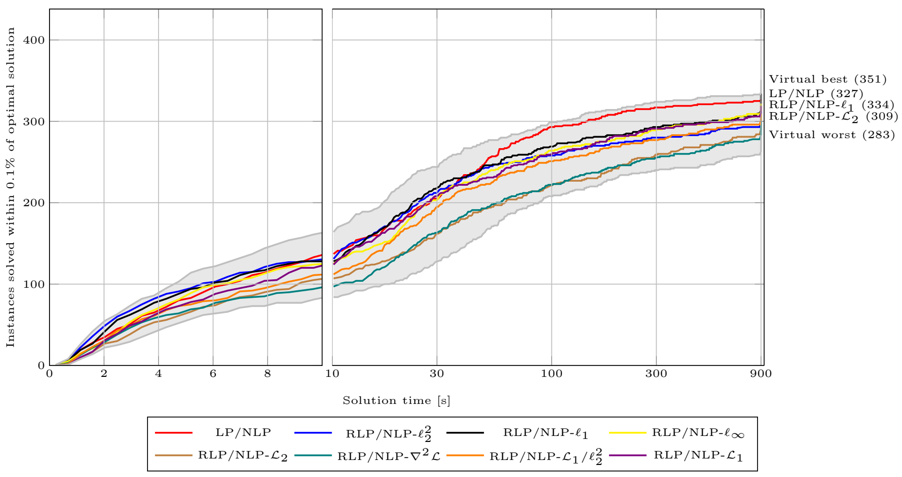

Fig. 13: Time performance profile for single-tree RLP/NLP methods as described in Algorithm 2.

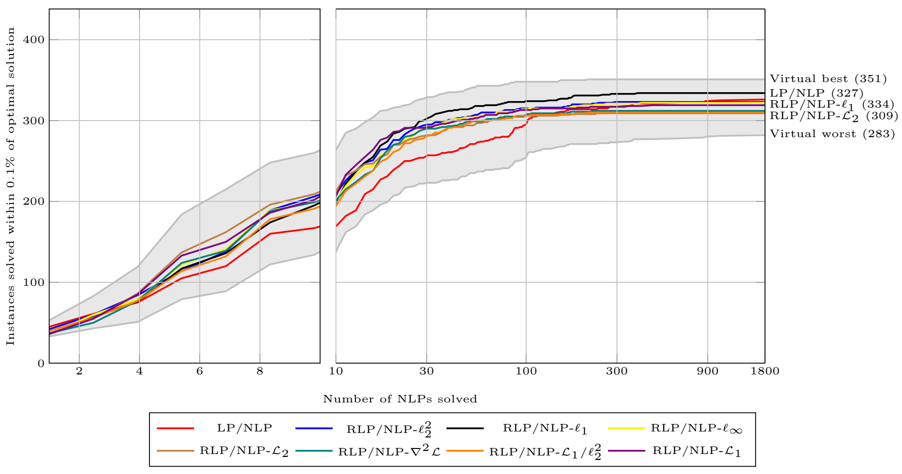

Fig. 14: Iteration performance profile single-tree RLP/NLP methods as described in Algorithm 2