Using Regularization and Second Order Information in Outer Approximation for Convex MINLP

Jan Kronqvist[a], David E. Bernal[b], and Ignacio E. Grossmann[b]

[a] *Process Design and Systems Engineering Åbo Akademi University, Turku, Finland*
[b] *Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh PA, USA*

December 14, 2017

Abstract

In this paper, we present two new methods for solving convex mixed-integer nonlinear programming problems based on the outer approximation method. The first method is inspired by the level method and uses a regularization technique to reduce the step size when choosing new integer combinations. The second method combines ideas from both the level method and the sequential quadratic programming technique and uses a second order approximation of the Lagrangean when choosing the new integer combinations. The main idea behind the methods is to choose the integer combination more carefully in each iteration, in order to obtain the optimal solution in fewer iterations compared to the original outer approximation method. We prove rigorously that both methods will find and verify the optimal solution in a finite number of iterations. Furthermore, we present a numerical comparison of the methods based on 109 test problems, which illustrates the benefits of the proposed methods.

1 Introduction

Mixed-integer nonlinear programming (MINLP) is a class of optimization problems containing both integer and continuous variables as well as nonlinear functions. The integer variables make it possible to incorporate logic relations and discrete quantities in the mathematical model. Together with linear and nonlinear constraints, MINLP becomes a powerful framework for modeling real-world optimization problems, and thus, there is a vast number of applications in areas such as engineering, computational chemistry, and finance [1, 2]. MINLP problems are by definition non-convex; however, they are still commonly

classified as either convex or non-convex. An MINLP problem is considered as convex if an integer relaxation results in a convex nonlinear programming (NLP) problem [3]. Convexity is a desirable property since it enables the direct use of several decomposition techniques for solving the problem. Such decomposition techniques are, *e.g.,* outer approximation (OA) [4], extended cutting plane (ECP) [5], extended supporting hyperplane (ESH) [6], generalized Benders decomposition (GBD) [7], and branch and bound (BB) techniques [8]. For reviews of MINLP methods and applications see [9, 3, 10, 11]. Even if there are several methods available for solving convex MINLP problems, it is still a challenging type of optimization problems as shown in the solver benchmark in [6].

Methods such as OA, ECP, ESH, and GBD all generate an iteratively improving linear approximation of the MINLP problem, where the nonlinear functions are underestimated by first-order Taylor series expansions. The linear approximation is a mixed-integer linear programming (MILP) problem and is often referred to as the MILP-master problem. All these methods iteratively choose the integer trial solutions as the minimizer of the MILP-master problem. Choosing the iterative solutions as the minimizer of a linear approximation is similar to the approach used in Kelley's method [12], which is an algorithm intended for convex NLP problems. It is known that Kelley's method is not efficient at handling nonlinearities and it has a poor complexity bound, *e.g.,* see [13]. Kelley's method is sometimes even referred to as unstable since the iterative solutions tend to make large jumps in the search space [14]. Since methods such as ECP, ESH, GBD, and OA choose the iterative integer solutions in the same manner as Kelley's method, they could also suffer from the same instability. Several techniques to reduce the instability of Kelley's method has successfully been used for NLP problems, *e.g.,* regularization to reduce the step size or the concept of a trust region [15].

Due to the non-convex nature of MINLP problems, it is not trivial to use regularization of the step size or a trust region when solving such problems, since the integer requirements may cause solutions to be far apart in the search space. However, recently there has been interest in the idea of using regularization for solving convex MINLP problems, *e.g.,* using quadratic stabilization with Benders decomposition was proposed in [16] and using regularization combined with a cutting plane method was presented in [17].

Here we present an approach for introducing stabilization in the subproblems for choosing the integer combination in OA. The stabilization technique is inspired by the regularization used in the level method for NLP, see [18, 19], and the method is referred to as level-based outer approximation (L-OA). By modifying the L-OA method it is possible to include second order information in the subproblems of choosing the integer combination, and we refer to this method as quadratic outer approximation (Q-OA). In Q-OA we use a second order Taylor series expansion for the Lagrangean function as the objective in the subproblems for finding a new integer combination. A similar quadratic approach was presented in [20]. However, the level constraint used in the level method provides a more robust way of enforcing an improvement and avoiding cycling. Furthermore, the level constraint forces the solutions to be chosen as

an interpolation between the minimizer of the Lagrangean approximation and the minimizer of the linear approximation in the MILP-master problem. The proposed methods are motivated by the strong convergence properties of the level method compared to Kelley's, and recent advances in software for solving MILP and mixed-integer quadratic programming (MIQP) problems.

The proposed methods are intended to accelerate the convergence of OA by choosing the integer combinations more carefully, using either a regularization technique or second-order information. Due to the regularization and the use of second-order derivatives, the proposed methods should be better at handling nonlinearities compared to OA. However, each iteration in L-OA and Q-OA will also be more complex than an iteration in OA. For MINLP problems with only a few nonlinear terms, there might not be significant improvements by the proposed methods. The methods are, thus, mainly intended for problems with moderate to high degree of nonlinearity. We begin with a brief review of OA in section 2, and from there we continue presenting the basics of L-OA and Q-OA in section 3 and 4. In section 5, it is proven that the convergence properties of OA still hold with the modifications in the proposed methods. Finally, in section 6 we present a numerical comparison of Q-OA, L-OA, and OA, based on test problems from the problem library MINLib2 [21].

2 Background

The MINLP problems considered here can be written as follows,

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

In order to guarantee global convergence, we need to assume some properties of the nonlinear functions. 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 define a nonempty compact set.

Assumption 3. For each feasible integer combination $\mathbf{y}$, an integer combination such that there exist $\mathbf{x}$ variables for which the problem is feasible, a constraint qualification holds, *e.g.,* Slater's condition [22].

These are the typical assumptions needed for rigorously proving convergence of OA, see [4, 20]. OA can be generalized to be applicable to non-differentiable problems, *e.g.,* see [23], although such problems are not considered here.

We begin by briefly presenting the main steps of the outer approximation method. As previously mentioned, the method uses a linear approximation of the MINLP problem to obtain trial solutions for the integer variables. Once an integer combination is obtained, the corresponding continuous variables can be determined by solving a continuous optimization problem. The previously obtained trial solutions $\left\{(\mathbf{x^i}, \mathbf{y^i})\right\}_{i=0}^{k}$ are used to construct the linear approximation of the MINLP problem. At iteration k, the next integer combination $\mathbf{y}^{k+1}$ is obtained by solving the following MILP subproblem

$$
\begin{aligned}
\min_{\mathbf{x},\mathbf{y},\mu} \quad & \mu \\
\text{s.t.} \quad & f(\mathbf{x^i}, \mathbf{y^i}) + \nabla f(\mathbf{x^i}, \mathbf{y^i})^T \begin{bmatrix} \mathbf{x} - \mathbf{x^i} \\ \mathbf{y} - \mathbf{y^i} \end{bmatrix} \leq \mu \quad \forall i = 1, \ldots, k, \\
& g_j(\mathbf{x^i}, \mathbf{y^i}) + \nabla g_j(\mathbf{x^i}, \mathbf{y^i})^T \begin{bmatrix} \mathbf{x} - \mathbf{x^i} \\ \mathbf{y} - \mathbf{y^i} \end{bmatrix} \leq 0 \quad \forall i = 1, \ldots k, \forall j \in A_i, \\
& \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} \leq \mathbf{b}, \\
& \mathbf{x} \in \mathbb{R}^n, \ \mathbf{y} \in \mathbb{Z}^m, \mu \in \mathbb{R}.
\end{aligned}
\tag{OA-master}
$$

Here A_i are index sets containing the indexes of the nonlinear constraint active at the trial solution $(\mathbf{x^i}, \mathbf{y^i})$ [20]. Due to convexity, we know that the feasible set is overestimated and that the objective will be underestimated, *e.g.,* see [4]. The optimum of problem (OA-master), thus, gives a valid lower bound to the MINLP problem, which is referred to as LB^{k+1}. Once the new integer combination $\mathbf{y^{k+1}}$ is obtained, the corresponding $\mathbf{x}$ variables can be obtained by solving the following convex NLP subproblem,

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

If problem (NLP-I) is feasible and solved to optimality, we obtain $\mathbf{x^{k+1}}$ and furthermore, the optimum provides a valid upper bound UB^{k+1} to the MINLP problem. Otherwise, if the NLP problem is infeasible we need a different approach to obtain the $\mathbf{x}$ variables and this can be done, for example, by solving a feasibility problem. The feasibility problem minimizes the constraint violation with the current choice of $\mathbf{y}$ variables, *e.g.,* using the ℓ_∞ norm, and it can be defined as,

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

By solving problem (NLP-f) the continuous variables $\mathbf{x^{k+1}}$ are obtained. However, in this case, $(\mathbf{x^{k+1}}, \mathbf{y^{k+1}})$ is not a feasible solution, and thus, no upper bound is obtained in this iteration. The feasibility problem always satisfies Slater's condition and due to the convexity assumption, we know that the feasibility problem is always feasible and tractable.

In case the difference between the upper and lower bound is not within the desired tolerance, we improve the linear approximation by adding new linearizations to problem (OA-master). These linearizations are often referred to as cutting planes or supporting hyperplanes, and they are given by,

$$\begin{aligned} f(\mathbf{x^{k+1}}, \mathbf{y^{k+1}}) + \nabla f(\mathbf{x^{k+1}}, \mathbf{y^{k+1}})^T \begin{bmatrix} \mathbf{x} - \mathbf{x^{k+1}} \\ \mathbf{y} - \mathbf{y^{k+1}} \end{bmatrix} \leq \mu, \\ g_j(\mathbf{x^{k+1}}, \mathbf{y^{k+1}}) + \nabla g_j(\mathbf{x^{k+1}}, \mathbf{y^{k+1}})^T \begin{bmatrix} \mathbf{x} - \mathbf{x^{k+1}} \\ \mathbf{y} - \mathbf{y^{k+1}} \end{bmatrix} \leq 0 \quad \forall j \in A_{k+1}. \end{aligned} \tag{1}$$

Due to convexity, the cuts will not exclude any feasible solution from the search space [24]. Adding these cuts to the MILP subproblem ensures that the integer combination $\mathbf{y^{k+1}}$ will not be obtained in a consecutive iteration unless it is the optimal integer solution. Convergence can be ensured since each iteration will either result in a new integer combination or verify optimality. For more details of OA see [4, 20, 25]. The basic steps of OA are summarized as a pseudo-code in Algorithm 1.

Here we have not considered the integer cuts used in [4], since these are not needed for convex problems. To get a better understanding of OA and to highlight the differences compared to the other methods, consider the following simple example

$$\begin{aligned} \text{minimize} \quad & -6x - y \\ \text{s.t.} \quad & 0.3(x-8)^2 + 0.04(y-6)^4 + 0.1e^{2x}y^{-4} \leq 56 \\ & 1/x + 1/y - x^{0.5}y^{0.5} \leq -4 \\ & 2x - 5y \leq -1 \\ & 1 \leq x \leq 20, \quad 1 \leq y \leq 20, \quad x \in \mathbb{R},\ y \in \mathbb{Z}. \end{aligned} \tag{Ex 1}$$

Later, we use the same example to illustrate the differences between the original OA and the proposed methods. To make the results comparable, we will use the starting point, $x^0 = 5.29$, $y^0 = 3$ with all the methods. Instead of solving the relaxed problem in the initialization step in Algorithm 1, we simply use (x^0, y^0) as a starting point. OA required 7 iterations to solve this problem, of which the first six iterations are shown in Figure 2. For this specific problem, the first four iterations all result in infeasible solutions where one of the nonlinear constraints are violated. As mentioned in Algorithm 1, the resulting solutions of the subproblems correspond to the optimal solution of (NLP-f). The optimal solution is obtained in iteration five, but verifying optimality requires two additional iterations.

Next, we will show how ideas from the level method can be combined with OA to get a stabilized approach for choosing new integer combinations.

Algorithm 1 An algorithm summarizing the basic steps of the outer approximation method

Define accepted optimality gap $\epsilon \geq 0$.

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-master).
   - 1.3 Set iteration counter $k = 1$, $UB^0 = \inf$ and $LB^0 = -\inf$.
2. Repeat until $UB^{k-1} - LB^{k-1} \leq \epsilon$.
   - 2.1 Solve problem (OA-master) to obtain $\mathbf{y^k}$ and LB^k
   - 2.4 Solve problem (NLP-I) with integer variables fixed as $\mathbf{y^k}$ to obtain $\mathbf{x^k}$.
     - 2.4.1 If problem (NLP-I) is infeasible, obtain $\mathbf{x^k}$ by solving feasibility problem (NLP-f) and set $UB^k = UB^{k-1}$.
   - 2.5 Generate cuts at $\mathbf{x^k}, \mathbf{y^k}$ according to (1) and add these to problems (OA-master).
   - 2.6 If $\mathbf{x^k}, \mathbf{y^k}$ is feasible, set $UB^k = \min\{f(\mathbf{x^k}, \mathbf{y^k}), UB^{k-1}\}$.
   - 2.7 Increase iteration counter, $k = k + 1$
3. Return the best found solution.

3 Level-based OA

The level method was originally presented in [18], as a method for solving non-smooth NLP problems. Like OA, the level method also constructs a linear approximation of the original optimization problem. However, the trial solutions are not chosen as the minimizer of the linear approximation. Instead, the trial solutions are obtained by projecting the current solution onto a specific level set of the linearly approximated objective function. For more details see [19, 13]. Here we will use a similar approach combined with OA, which we show is equivalent to adding specific trust regions to the problems (OA-master) in the original OA.

Here we assume that a feasible solution to the MINLP problem $\bar{\mathbf{x}}, \bar{\mathbf{y}}$ is known. Such a solution can for example be obtained by first preforming some original OA iterations or by using a specific procedure such as the feasibility pump [26]. An upper bound to the MINLP problem is, thus, given by $f(\bar{\mathbf{x}}, \bar{\mathbf{y}})$ and cuts at $\bar{\mathbf{x}}, \bar{\mathbf{y}}$ can be generated according to (1) to form problem (OA-master). A valid lower bound LB^1 can be obtained by solving the linear subproblem

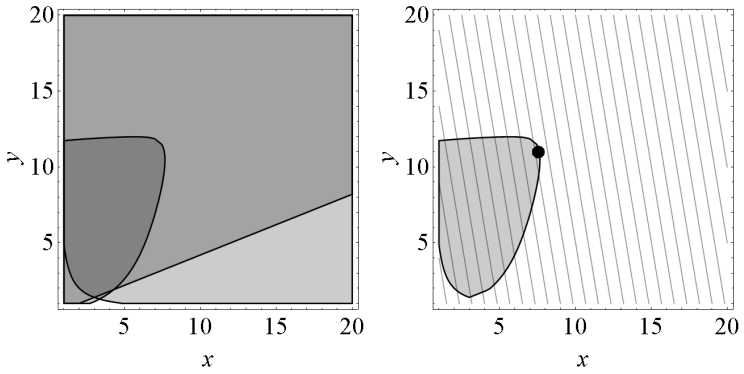

Figure 1: The figure to the left shows the feasible regions of the constraints in problem (Ex 1). The second figure shows the integer relaxed feasible region, contours of the objective and the optimal solution.

(OA-master), and thus we have bounds for the optimal solution f^*, *i.e.,* $LB^1 \leq f^* \leq f(\bar{\mathbf{x}}, \bar{\mathbf{y}})$.

From the bounds of the optimal solution we can in each iteration k estimate a value of the optimal solution according to,

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

where $\alpha \in (0, 1]$ and $\bar{\mathbf{x}}, \bar{\mathbf{y}}$ is chosen as the best found feasible solution, similarly as in the level method. The lower bound LB^k is obtained as in the original OA, by solving problem (OA-master). In eq. (2) α is a parameter which states how much we trust the linear approximation of the MINLP problem. Setting α close to one results in an estimated optimum $\hat{f}_k^*$ close to the lower bound, while setting it close to zero results in an estimated optimum close to the best incumbent solution. The next integer solution $\mathbf{y^{k+1}}$ can now be obtained by projecting $\bar{\mathbf{x}}, \bar{\mathbf{y}}$ onto the $\hat{f}_k^*$ level set of the linearly approximated objective function. The projection is performed by solving the following MIQP problem,

$$\begin{aligned}
\min_{\mathbf{x},\mathbf{y},\mu} \quad & \left\| \begin{matrix} \mathbf{x} - \bar{\mathbf{x}} \\ \mathbf{y} - \bar{\mathbf{y}} \end{matrix} \right\|^2 \\
\text{s.t.} \quad & \mu \leq \hat{f}_k^* \\
& f(\mathbf{x^i}, \mathbf{y^i}) + \nabla f(\mathbf{x^i}, \mathbf{y^i})^T \begin{bmatrix} \mathbf{x} - \mathbf{x^i} \\ \mathbf{y} - \mathbf{y^i} \end{bmatrix} \leq \mu \quad \forall i = 1, \ldots, k, \\
& g_j(\mathbf{x^i}, \mathbf{y^i}) + \nabla g_j(\mathbf{x^i}, \mathbf{y^i})^T \begin{bmatrix} \mathbf{x} - \mathbf{x^i} \\ \mathbf{y} - \mathbf{y^i} \end{bmatrix} \leq 0 \quad \forall i = 1, \ldots k, \forall j \in A_i, \\
& \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} \leq \mathbf{b}, \\
& \mathbf{x} \in \mathbb{R}^n, \ \mathbf{y} \in \mathbb{Z}^m, \mu \in \mathbb{R},
\end{aligned} \tag{MIQP-Proj}$$

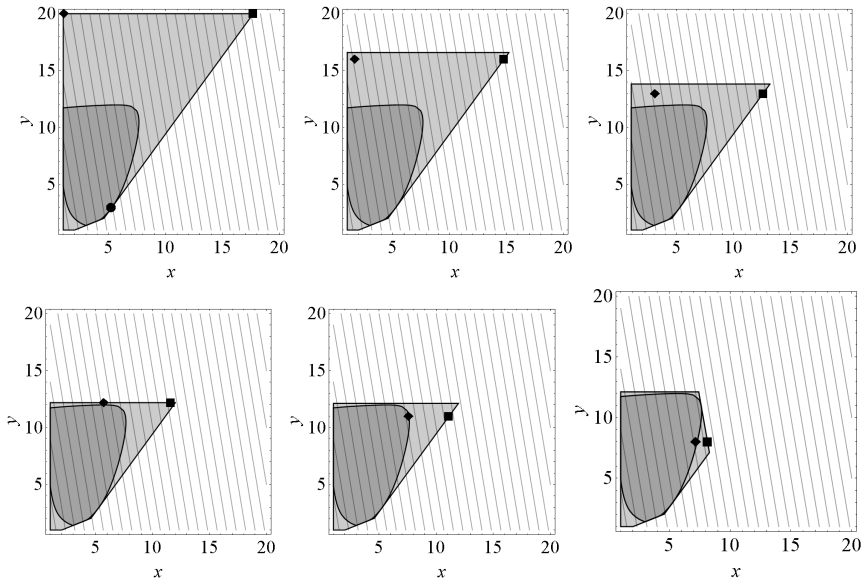

Figure 2: The figures show the feasible region defined by the nonlinear constraints in dark gray, and the light gray areas show the outer approximation obtained by the generated cuts. The squared dots represent the solutions obtained from the MILP subproblem and diamond shaped dots represent the solutions obtained by one of the NLP subproblems. The dot in the first figure shows the starting point (x^0, y^0).

where $\|\cdot\|$ is the Euclidean norm. The MIQP problem should contain all the supporting hyperplanes and cutting planes present in problem (OA-master), which was solved to obtain the lower bound. The next integer solution $\mathbf{y^{k+1}}$ is thereby chosen as a point as close as possible to the best known feasible solution which reduce the linearly approximated objective to at most $\hat{f}_k^*$. Since $\hat{f}_k^*$ is calculated according to (2) there always exists a solution to the MIQP problem, *e.g.,* the minimizer of problem (OA-master) will satisfy all the constraints. Once the new integer combination is obtained, the corresponding continuous variables can be determined using the same technique as described in the previous section. We summarize the level based outer approximation as a pseudo-code in Algorithm 2.

The difference compared to the original OA is the two-step procedure for obtaining the new integer combination, which involves both the solution of an MILP and an MIQP subproblem. This increases the complexity at each iteration. However, as we will prove later the MIQP need not to be solved to optimality. Basically, any feasible solution to the MIQP will be sufficient for ensuring convergence. The computational aspects are described in more detail in section 6 and convergence of both L-OA and Q-OA is proved in section 5.

To obtain a geometrical understanding of how L-OA differs to the original

Algorithm 2 An algorithm summarizing the basic steps of level-based outer approximation (L-OA)

Define accepted optimality gap $\epsilon \geq 0$ and choose the parameter $\alpha \in (0, 1]$.

1. Initialization.
   - 1.1 Obtain a feasible solution $\bar{\mathbf{x}}, \bar{\mathbf{y}}$, either by OA or by any other technique.
   - 1.2 Generate cuts at $\bar{\mathbf{x}}, \bar{\mathbf{y}}$ according to (1) and construct problems (OA-master) and (MIQP-Proj).
   - 1.3 Set iteration counter $k = 1$, and $LB^0 = -\inf$.
2. Repeat until $f(\bar{\mathbf{x}}, \bar{\mathbf{y}}) - LB^{k-1} \leq \epsilon$.
   - 2.1 Solve problem (OA-master) to obtain LB^k
   - 2.2 Calculate the estimated optimal value $\hat{f}_k^*$ according to (2).
   - 2.3 Solve problem (MIQP-Proj) to obtain $\mathbf{y^k}$
   - 2.4 Solve problem (NLP-I) with integer variables fixed as $\mathbf{y^k}$ to obtain $\mathbf{x^k}$.
     - 2.4.1 If problem (NLP-I) is infeasible, obtain $\mathbf{x^k}$ by solving feasibility problem (NLP-f).
   - 2.5 Generate cuts at $\mathbf{x^k}, \mathbf{y^k}$ according to (1) and add these to problems (OA-master) and (MIQP-Proj).
   - 2.6 If $\mathbf{x^k}, \mathbf{y^k}$ is feasible and $f(\mathbf{x^k}, \mathbf{y^k}) \leq f(\bar{\mathbf{x}}, \bar{\mathbf{y}})$, set $\bar{\mathbf{x}}, \bar{\mathbf{y}} = \mathbf{x^k}, \mathbf{y^k}$.
   - 2.7 Increase iteration counter, $k = k + 1$
3. Return $\bar{\mathbf{x}}, \bar{\mathbf{y}}$ as the optimal solution.

OA, we again consider problem (Ex 1). Here we use the same starting point as before and we set the level parameter as $\alpha = 0.4$. To solve the problem with these parameters L-OA requires four iterations. The three first iterations are shown in Figure 3. In the fourth iteration, we are able to verify optimality directly after solving the MILP subproblem since we obtain $LB^4 = f(\bar{\mathbf{x}}, \bar{\mathbf{y}})$.

As mentioned earlier, L-OA will find similar integer solutions as adding specific trust regions to the MILP subproblems in the original OA. This property is further described in Theorem 1.

Theorem 1. *The procedure of solving problems* (OA-master) *and* (MIQP-Proj) *will result in a solution equivalent to adding the trust region constraint*

$$\left\| \begin{matrix} \mathbf{x} - \bar{\mathbf{x}} \\ \mathbf{y} - \bar{\mathbf{y}} \end{matrix} \right\|^2 \leq r_k, \tag{3}$$

to problem (OA-master) *in the original OA, where* r_k *i chosen as the optimum*

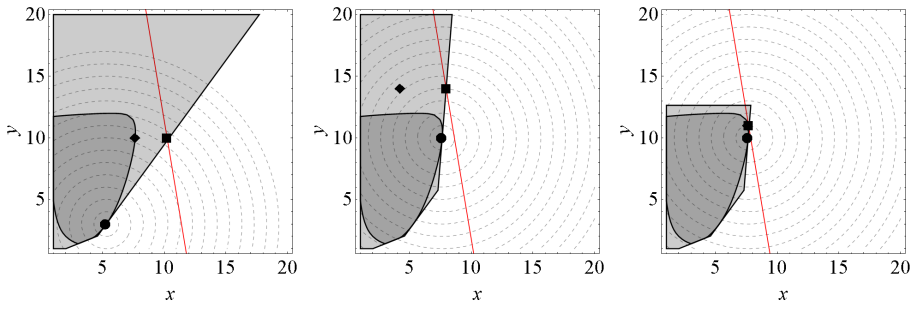

Figure 3: The figure illustrates the first three iterations needed to solve problem (Ex 1) with the L-OA method. The dashed circles represent the contours of the objective function in the MIQP subproblems and the red line shows the level constraint given by $\mu \leq \hat{f}_k^*$. The circular dots represent the best-found solution so far, the squared dots represent the solutions obtained from the MIQP subproblem and diamond shaped dots represent the solutions obtained by one of the NLP subproblems.

of problem (MIQP-Proj).

Proof. First, assume that there exists a unique solution to problem (MIQP-Proj), and denote the minimizer as $\mathbf{x}^{\text{MIQP}}, \mathbf{y}^{\text{MIQP}}, \mu^{\text{MIQP}}$. As stated the radius of the trust region constraint is chosen as

$$r_k = \left\| \begin{matrix} \mathbf{x}^{\text{MIQP}} - \bar{\mathbf{x}} \\ \mathbf{y}^{\text{MIQP}} - \bar{\mathbf{y}} \end{matrix} \right\|^2 \tag{4}$$

Adding the trust region constraint given by eq. (3) with radius r_k to problem (OA-master) gives the solution $\mathbf{x}^{\text{MILP}}, \mathbf{y}^{\text{MILP}}, \mu^{\text{MILP}}$. Now, assume this solution is not the same as the MIQP solution. Since the MIQP solution is assumed to be unique and not equal to the MILP solution, it follows that,

$$r_k > \left\| \begin{matrix} \mathbf{x}^{\text{MILP}} - \bar{\mathbf{x}} \\ \mathbf{y}^{\text{MILP}} - \bar{\mathbf{y}} \end{matrix} \right\|^2 . \tag{5}$$

Furthermore, since (OA-master) minimizes μ we get $\mu^{\text{MILP}} \leq \mu^{\text{MIQP}} \leq \hat{f}_k^*$. This leads to a contradiction since $\mathbf{x}^{\text{MILP}}, \mathbf{y}^{\text{MILP}}, \mu^{\text{MILP}}$ would then define a feasible solution to problem (MIQP-Proj) with an objective strictly lower than the solution obtained by solving the minimization problem.

In case there is not a unique solution to problem (MIQP-Proj), then all these solutions are also optimal solutions to problem (OA-master) with the trust region constraint. To prove the statement, assume $\mathbf{x}^{\text{MILP}}, \mathbf{y}^{\text{MILP}}, \mu^{\text{MILP}}$ is not an optimal solution to problem (MIQP-Proj). As assumed the MILP solution is not an optimal solution to problem (MIQP-Proj) and it satisfies the trust region constraint given by eq. (3). Therefore, eq. (5) must hold with

strict inequality. This leads to the same contradiction as in the case of a unique solution. □

Note that there are no practical implications that follow from Theorem 1 because the radius of the trust region resulting in similar solutions cannot be determined in advance. However, the Theorem shows that the procedure used in L-OA can be viewed as a technique of using a trust region with OA. Next, we show that it is possible to use a similar approach as L-OA to incorporate second order information in the task of obtaining the integer combinations.

4 Quadratic outer approximation

In order to obtain better integer solutions, it would be desirable to use information regarding the curvature of the constraints and objective in the task of choosing the integer combinations. Here we propose a technique where second-order information is incorporated by minimizing a second order Taylor series expansion of the Lagrangean function. By using the Lagrangean it is possible to include curvature of both the constraints and objective while keeping the constraints of the subproblems linear.

Here we define the Lagrangean function $\mathcal{L} : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^l \to \mathbb{R}$ as

$$\mathcal{L}(\mathbf{x}, \mathbf{y}, \lambda) = f(\mathbf{x}, \mathbf{y}) + \sum_{j=1}^{l} \lambda_j g_j(\mathbf{x}, \mathbf{y}), \tag{6}$$

where $\lambda_j \geq 0$ is the Lagrange multiplier of the j-th nonlinear constraint. Here we do not include the linear constraints in the Lagrangean, since these are handled directly in the subproblems. The Lagrangean is frequently used in NLP techniques and has the following important properties

Property 1. If all nonlinear functions $f, g_1, \ldots, g_j$ in problem (MINLP) are convex, then with nonnegative multipliers the Lagrangean defined in eq. (6) will be a convex function in the $\mathbf{x}, \mathbf{y}$ variables, *e.g.,* see [27, 24].

Property 2. Strong duality holds for convex optimization problems that satisfy Slater's condition; *i.e.,* there exists valid multipliers such that the minimum of the Lagrangean is equal to the minimum of the original problem [24].

Since the MINLP problems are non-convex by nature, we cannot expect strong duality to hold. However, the first property is important since it will ensure that the subproblem we use for finding the integer combinations will be tractable. We do not want to directly minimize the Lagrangean, because, that problem is basically as difficult as the original problem. Therefore, we will use a second order approximation of the Lagrangean, which is given by

$$\mathcal{L}(\bar{\mathbf{x}}, \bar{\mathbf{y}}, \bar{\lambda}) + \nabla_{\mathbf{x},\mathbf{y}} \mathcal{L}(\bar{\mathbf{x}}, \bar{\mathbf{y}}, \bar{\lambda})^T \begin{bmatrix} \Delta\mathbf{x} \\ \Delta\mathbf{y} \end{bmatrix} + \frac{1}{2} \begin{bmatrix} \Delta\mathbf{x} \\ \Delta\mathbf{y} \end{bmatrix}^T \nabla^2_{\mathbf{x},\mathbf{y}} \mathcal{L}(\bar{\mathbf{x}}, \bar{\mathbf{y}}, \bar{\lambda}) \begin{bmatrix} \Delta\mathbf{x} \\ \Delta\mathbf{y} \end{bmatrix}, \tag{7}$$

where $\nabla_{\mathbf{x},\mathbf{y}}\mathcal{L}$ is the gradient of the Lagrangean with respect to $\mathbf{x}, \mathbf{y}$ and $\nabla^2_{\mathbf{x},\mathbf{y}}$ denotes the Hessian matrix. To make the notation more compact we have introduced the Δ-variables that are given by $\Delta\mathbf{x} = \mathbf{x} - \bar{\mathbf{x}}$ and $\Delta\mathbf{y} = \mathbf{y} - \bar{\mathbf{y}}$. Due to Property 1, we know that that the Hessian $\nabla^2_{\mathbf{x},\mathbf{y}}$ will be positive semidefinite for all $\lambda \geq \mathbf{0}$. For small changes in the Δ-variables eq. (7) should give a good approximation, although the approximation does not under or over estimate the real Lagrangean function.

The natural approach of using the quadratic approximation in OA would be to replace the linear objective of the MILP-master problem by the quadratic function given by eq. (7). However, this approach will not work on its own because the second order approximation does not necessarily underestimate the Lagrangean. Therefore, it is possible that the approximation point $\bar{\mathbf{x}}, \bar{\mathbf{y}}$ is the optimum of the approximation even if it is not the optimal solution to the original problem, and thus, it can stagnate at non-optimal solutions. To avoid this, the method presented in [20] uses an ϵ improvement strategy, where the next solution must reduce the linearly approximated objective by a small ϵ-value. The ϵ improvement is enforced by the following constraints

$$\begin{aligned} &\mu \leq f(\bar{\mathbf{x}}, \bar{\mathbf{y}}) - \epsilon \\ &f(\mathbf{x^i}, \mathbf{y^i}) + \nabla f(\mathbf{x^i}, \mathbf{y^i})^T \begin{bmatrix} \mathbf{x} - \mathbf{x^i} \\ \mathbf{y} - \mathbf{y^i} \end{bmatrix} \leq \mu \quad \forall i = 1, \ldots, k, \end{aligned} \tag{8}$$

where $\bar{\mathbf{x}}, \bar{\mathbf{y}}$ is the best found solution. With this approach ϵ must be chosen smaller than the desired optimality gap. Thus, it will only result in a small reduction requirement. Therefore, the quadratic outer approximation method in [20] will rely heavily on the second order approximation of the Lagrangean. In case the approximation point $\bar{\mathbf{x}}, \bar{\mathbf{y}}$ with the corresponding multipliers $\bar{\lambda}$ is not the optimal solution to the MINLP, then the Lagrangean might not give a good approximation of the original problem and this might cause slow convergence. Due to the discrete nature of MINLP problems, it is possible that only the optimal integer combination with the corresponding continuous variables will result in the optimal set of active constraints and nonzero multipliers.

Here we use a different approach, which combines information from both the linear approximation with the quadratic approximation of the Lagrangean, to make sure the proposed method does not stagnate at non-optimal solutions. By using the same approach as in L-OA, an estimate of the optimal solution $\hat{f}^*_k$ can be calculated according to eq. (2). The estimated optimum can further be used to construct the following reduction constraint,

$$\begin{aligned} &\mu \leq \hat{f}^*_k \\ &f(\mathbf{x^i}, \mathbf{y^i}) + \nabla f(\mathbf{x^i}, \mathbf{y^i})^T \begin{bmatrix} \mathbf{x} - \mathbf{x^i} \\ \mathbf{y} - \mathbf{y^i} \end{bmatrix} \leq \mu \quad \forall i = 1, \ldots, k. \end{aligned} \tag{9}$$

As long as $\hat{f}^*_k$ is calculated using the same technique as in L-OA there will always exist a solution that satisfies the reduction constraints in eq. (9). Furthermore, since $\hat{f}^*_k$ is chosen as an interpolation between the upper and lower bound it

will usually result in a stricter reduction constraint. We will now construct the master problem by minimizing the quadratic approximation of the Lagrangean with the reduction constraint given by eq. (9), the accumulated cuts given by eq. (1) and all linear constraints from the MINLP problem. The new integer combination y^{k+1} is, thus, obtain by solving the following MIQP problem,

$$
\begin{aligned}
\min_{\mathbf{x},\mathbf{y},\mu} \quad & \nabla_{\mathbf{x},\mathbf{y}}\mathcal{L}(\bar{\mathbf{x}},\bar{\mathbf{y}},\bar{\lambda})^T \begin{bmatrix}\Delta\mathbf{x}\\ \Delta\mathbf{y}\end{bmatrix} + \frac{1}{2}\begin{bmatrix}\Delta\mathbf{x}\\ \Delta\mathbf{y}\end{bmatrix}^T \nabla^2_{\mathbf{x},\mathbf{y}}\mathcal{L}(\bar{\mathbf{x}},\bar{\mathbf{y}},\bar{\lambda})\begin{bmatrix}\Delta\mathbf{x}\\ \Delta\mathbf{y}\end{bmatrix} \\
\text{s.t.} \quad & \mu \leq \hat{f}_k^* \\
& f(\mathbf{x^i},\mathbf{y^i}) + \nabla f(\mathbf{x^i},\mathbf{y^i})^T \begin{bmatrix}\mathbf{x}-\mathbf{x^i}\\ \mathbf{y}-\mathbf{y^i}\end{bmatrix} \leq \mu \quad \forall i = 1,\ldots,k \\
& g_j(\mathbf{x^i},\mathbf{y^i}) + \nabla g_j(\mathbf{x^i},\mathbf{y^i})^T \begin{bmatrix}\mathbf{x}-\mathbf{x^i}\\ \mathbf{y}-\mathbf{y^i}\end{bmatrix} \leq 0 \quad \forall i = 1,\ldots k, \forall j \in A_i, \\
& \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} \leq \mathbf{b}, \\
& \mathbf{x} \in \mathbb{R}^n,\ \mathbf{y} \in \mathbb{Z}^m, \mu \in \mathbb{R},
\end{aligned}
\tag{QOA-master}
$$

where $\Delta\mathbf{x} = \mathbf{x} - \bar{\mathbf{x}}$ and $\Delta\mathbf{y} = \mathbf{y} - \bar{\mathbf{y}}$. As in L-OA $\bar{\mathbf{x}}, \bar{\mathbf{y}}$ is chosen as the best found feasible solution and $\bar{\lambda}$ are the corresponding Lagrangean multipliers obtained by solving problem (NLP-I). The NLP subproblem with fixed integer variables will provide both the $\mathbf{x}$ variables and the multipliers λ. If the NLP subproblem is infeasible we solve the problem (NLP-f), from which we obtain the corresponding multipliers. As mentioned before $\nabla^2_{\mathbf{x},\mathbf{y}}$ is positive semidefinite due to the convexity of the nonlinear functions; therefore, the MIQP problem can be solved efficiently with software such as Gurobi[28] or Cplex[29].

Once the next integer solution has been obtained, the continuous variables are determined as in OA or L-OA, and more cuts are generated according to eq. (1). The lower bound is updated in each iteration as in L-OA by solving problem (OA-master). The quadratic outer approximation method is summarized as a pseudocode in Algorithm 3.

As in L-OA, each iteration includes both an MILP and an MIQP subproblem. We will show later that it is sufficient to merely obtain a feasible solution to the MIQP, which can reduce the computational complexity of both the L-OA and Q-OA method. Section 5 proves the method's convergence to the optimal solution in a finite number of iterations, and Section 6 discusses the computational aspect more in detail.

The technique used for obtaining the integer combinations in Q-OA actually results in an interpolation between the minimizer of the linear approximation in problem (OA-master) and the minimizer of the Lagrangean approximation, where α in eq. (2) is the interpolation parameter. Setting $\alpha = 1$ will force the solution of problem (QOA-master) to the minimizer of problem (OA-master), and setting α close to zero will allow the solution to be close to the minimizer of the Lagrangean approximation. The Q-OA method will, therefore, be less sensitive to the accuracy of the Lagrangean approximation, compared to the

Algorithm 3 An algorithm summarizing the basic steps of the quadratic outer approximation (Q-OA) method

Define accepted optimality gap $\epsilon \geq 0$ and choose the parameter $\alpha \in]0, 1]$.

1. Initialization.
   - 1.1 Obtain a feasible solution $\bar{\mathbf{x}}, \bar{\mathbf{y}}$ and the multipliers $\bar{\lambda}$, either by OA or by any other technique.
   - 1.2 Generate cuts at $\bar{\mathbf{x}}, \bar{\mathbf{y}}$ according to (1) and construct problems (OA-master) and (QOA-master).
   - 1.3 Set iteration counter $k = 1$, and $LB^0 = -\inf$.
2. Repeat until $f(\bar{\mathbf{x}}, \bar{\mathbf{y}}) - LB^{k-1} \leq \epsilon$.
   - 2.1 Solve problem (OA-master) to obtain LB^k
   - 2.2 Calculate the estimated optimal value $\hat{f}_k^*$ according to (2).
   - 2.3 Solve problem (QOA-master) to obtain $\mathbf{y^k}$
   - 2.4 Solve problem (NLP-I) with integer variables fixed as $\mathbf{y^k}$ to obtain $\mathbf{x^k}$ and $\lambda^{\mathbf{k}}$.
     - 2.4.1 If problem (NLP-I) is infeasible, obtain $\mathbf{x^k}$ by solving feasibility problem (NLP-f).
   - 2.5 Generate cuts at $\mathbf{x^k}, \mathbf{y^k}$ according to (1) and add these to problems (OA-master) and (QOA-master).
   - 2.6 If $\mathbf{x^k}, \mathbf{y^k}$ is feasible and $f(\mathbf{x^k}, \mathbf{y^k}) \leq f(\bar{\mathbf{x}}, \bar{\mathbf{y}})$, set $\bar{\mathbf{x}}, \bar{\mathbf{y}}, \bar{\lambda} = \mathbf{x^k}, \mathbf{y^k}, \lambda^{\mathbf{k}}$.
   - 2.7 Increase iteration counter, $k = k + 1$
3. Return $\bar{\mathbf{x}}, \bar{\mathbf{y}}$ as the optimal solution.

method in [20]. In the next section, we prove that the finite convergence of Q-OA can still be guaranteed even if the Hessian of the Lagrangean is only estimated as long as it remains positive semidefinite.

To provide a geometric interpretation of the method and to show how it differs from OA and L-OA, we apply the method to the illustrative test problem (Ex 1). We use the same starting point (x^0, y^0) as before and we set the level parameter as $\alpha = 0.5$. To solve the problem with these parameters Q-OA requires three iterations. The first two iterations are shown are shown in Figure 3. In the third iteration, we are able to verify optimality after only solving the MILP subproblem, since we obtain $LB^3 = f(\bar{\mathbf{x}}, \bar{\mathbf{y}})$. From the figure, note that the reduction constraint given by eq. (9), prevents the algorithm form taking a too short step in the first iteration and the optimal solution is actually obtained in the first iteration. If the trial solution had only been chosen as the minimizer of the Lagrangean relaxation, it would have resulted in less progress per itera-

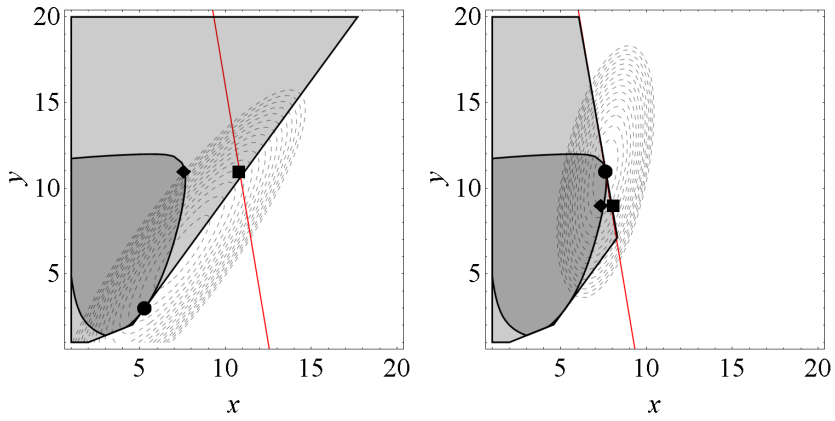

Figure 4: The figures illustrate the first two iterations needed to solve problem (Ex 1) with the Q-OA method. The dashed eclipses represent the contours of the approximated Lagrangean used as objective in the MIQP subproblem and the red line shows the level constraint given by $\mu \leq \hat{f}_k^*$. The circular dots represent the best found solution so far, the squared dots represent the solutions obtained from the MIQP subproblem and diamond shaped dots represent the solutions obtained by one of the NLP subproblems.

tion. It should also be noted that not a single infeasible integer combination was encountered.

5 Convergence properties

Proving finite convergence of L-OA and Q-OA can be done quite similarly as for the original OA, and some of the results from [4, 20] are directly applicable. Finite convergence can be proven as follows. We show that an infeasible integer combination obtained by L-OA or Q-OA will be cut off by the cuts generated according to eq. (1) and therefore, this integer combination cannot be obtained in any future iteration. Next, we prove that a specific integer combination cannot be obtained twice with either method, unless optimality is proven. The methods, therefore, obtain new integer combinations in each iteration, and since there are only a finite number of such combinations, the methods will converge in a finite number of iteration.

Convexity of the nonlinear functions is crucial here since it ensures that no feasible integer solution is cut off by the cuts generated by L-OA or Q-OA and that problem (OA-master) gives a valid lower bound, as is stated in Lemma 1.

Lemma 2. *Solving problem* (OA-master) *will give a valid lower bound to the optimum of the MINLP problem.*

Proof. From the first order convexity condition we know that for any convex

differentiable function $\phi(\mathbf{x}, \mathbf{y})$,

$$\phi(\mathbf{x}, \mathbf{y}) \geq \phi(\mathbf{x^0}, \mathbf{y^0}) + \nabla\phi(\mathbf{x^0}, \mathbf{y^0})^T \begin{bmatrix} \mathbf{x} - \mathbf{x^0} \\ \mathbf{y} - \mathbf{y^0} \end{bmatrix} \quad \forall (\mathbf{x}, \mathbf{y}), (\mathbf{x^0}, \mathbf{y^0}) \in D_\phi,$$

where D_ϕ is the domain in which the function is convex. Therefore the feasible region of the problem (MINLP) will be overestimated and the objective function will be underestimated at each iteration. □

Lemma 3. *An infeasible integer combination* $\mathbf{y^k}$, i.e., *an integer combination such that problem* (NLP-I) *is infeasible, will be cut off by the cuts generated in L-OA and Q-OA.*

Proof. It is proved in [20], that solving the feasibility problem and adding cuts for the active constraints will cut off $\mathbf{y^k}$ from the search space. For more details see [20] Lemma 1 page 331. □

Lemma 4. *If the lower bound is not equal to the upper bound, then there exists a solution to the MIQP subproblems in L-OA and Q-OA.*

Proof. Due to convexity, the linearly approximated problem (OA-master) will always be feasible if the MINLP problem is feasible. The MIQP subproblem in both L-OA and Q-OA contains the same constraints as problem (OA-master) and the reduction constraint. Since $\hat{f}_k^*$ is calculated according to eq. (2), the solution to problem (OA-master) is a feasible solution to the MIQP subproblem. In case problem (OA-master) is infeasible, the search is terminated and MIQP will not be solved since it verifies that the MINLP is infeasible. □

Theorem 5. *If the lower bound is not equal to the upper bound, then the MIQP subproblems in L-OA and Q-OA will give a new integer combination.*

Proof. By Lemma 2, we know that all infeasible integer combination that has been found are cut off from the search space by the cuts added to the subproblems. Since the upper and lower bound are not equal, we know that the estimated optimum will be smaller than the upper bound, *i.e.,* $\hat{f}_k^* < f(\bar{\mathbf{x}}, \bar{\mathbf{y}})$. This is obviously true for all feasible solutions found so far, which we denote as $\hat{\mathbf{x}}^\mathbf{i}, \hat{\mathbf{y}}^\mathbf{i}$, and the following relation is obtained,

$$\hat{f}_k^* < f(\bar{\mathbf{x}}, \bar{\mathbf{y}}) \leq f(\hat{\mathbf{x}}^\mathbf{i}, \hat{\mathbf{y}}^\mathbf{i}) \quad \forall i. \tag{10}$$

At all the obtained feasible solutions $\hat{\mathbf{x}}^\mathbf{i}, \hat{\mathbf{y}}^\mathbf{i}$ the methods generate the following linearizations of the objective,

$$f(\hat{\mathbf{x}}^\mathbf{i}, \hat{\mathbf{y}}^\mathbf{i}) + \nabla f(\hat{\mathbf{x}}^\mathbf{i}, \hat{\mathbf{y}}^\mathbf{i})^T \begin{bmatrix} \mathbf{x} - \hat{\mathbf{x}}^\mathbf{i} \\ \mathbf{y} - \hat{\mathbf{y}}^\mathbf{i} \end{bmatrix} \leq \mu. \tag{11}$$

The minimum value of μ at these feasible solutions, denoted $\hat{\mu}^i$, will therefore satisfy the following relation,

$$\hat{f}_k^* < f(\bar{\mathbf{x}}, \bar{\mathbf{y}}) \leq \hat{\mu}^i \quad \forall i. \tag{12}$$

In both the MIQP subproblem in L-OA and Q-OA, we have the reduction constraint $\mu \leq \hat{f}_k^*$, and from eq. (11) it follows that the next solution must satisfy,

$$\nabla f(\hat{\mathbf{x}}^\mathbf{i}, \hat{\mathbf{y}}^\mathbf{i})^T \begin{bmatrix} \mathbf{x} - \hat{\mathbf{x}}^\mathbf{i} \\ \mathbf{y} - \hat{\mathbf{y}}^\mathbf{i} \end{bmatrix} < 0 \quad \forall i. \tag{13}$$

Now, assume that one of the obtained feasible solutions $\tilde{\mathbf{x}}, \tilde{\mathbf{y}} \in \{\hat{\mathbf{x}}^\mathbf{i}, \hat{\mathbf{y}}^\mathbf{i}\}$ can be perturbed in the $\mathbf{x}$-variables by $\Delta\mathbf{x}$ such that it satisfies all constraints of the MIQP subproblem and the property given by eq. (13). Since $\tilde{\mathbf{x}}$ was obtained by solving problem (NLP-I), it must satisfy the KKT-conditions,

$$\begin{aligned} &\nabla_\mathbf{x} f(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}) + \sum_{j=1}^{l} \lambda_j \nabla_\mathbf{x} g_j(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}) + \mathbf{A}^T \gamma = \mathbf{0} \\ &g_j(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}) \leq 0 \quad \forall j = 1, \ldots, l \\ &\mathbf{A}\tilde{\mathbf{x}} + \mathbf{B}\tilde{\mathbf{y}} \leq \mathbf{b} \\ &\lambda, \gamma \geq \mathbf{0} \\ &\lambda_j g_j(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}) = 0 \quad \forall j = 1, \ldots, l \\ &(\mathbf{A}\tilde{\mathbf{x}} + \mathbf{B}\tilde{\mathbf{y}} - \mathbf{b}) \circ \gamma = \mathbf{0}, \end{aligned} \tag{14}$$

where $\nabla_\mathbf{x}$ is the gradient with respect to $\mathbf{x}$-variables, and γ are the multipliers of the linear constraints. At the solution $\tilde{\mathbf{x}}, \tilde{\mathbf{y}}$, the methods will generate the following supporting hyperplanes,

$$g_j(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}) + \nabla g_j(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})^T \begin{bmatrix} \mathbf{x} - \tilde{\mathbf{x}} \\ \mathbf{y} - \tilde{\mathbf{y}} \end{bmatrix} \leq 0 \quad \forall j \mid \lambda_j \neq 0. \tag{15}$$

Since these are all active constraints, the constant on the left hand side must be zero, *i.e.,* $g_j(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}) = 0$. The perturbation $\Delta\mathbf{x}$ must satisfy eq. (15), which can be written as ,

$$\lambda_j \nabla_x g_j(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})^T \Delta\mathbf{x} \leq 0 \quad \forall j = 1, \ldots, l. \tag{16}$$

The same is also true for the linear constraints. For all active linear constraints $\Delta\mathbf{x}$ cannot increase the value of the left hand side. This condition can be summed over all linear constraints by the multipliers γ as,

$$\gamma^T \mathbf{A} \Delta\mathbf{x} \leq 0. \tag{17}$$

The perturbation also has to satisfy the reduction stated in eq. (13), which yields,

$$\nabla_x f(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})^T \Delta\mathbf{x} < 0. \tag{18}$$

Adding all inequalities from eq. (16), (17) and (18) results in the following strict inequality,

$$\nabla_\mathbf{x} f(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})^T \Delta\mathbf{x} + \sum_{j=1}^{l} \lambda_j \nabla_\mathbf{x} g_j(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})^T \Delta\mathbf{x} + \gamma^T \mathbf{A} \Delta\mathbf{x} < 0. \tag{19}$$

However, taking the inner product of $\Delta\mathbf{x}$ and both sides of the first KKT condition results in the following equality

$$\nabla_{\mathbf{x}} f(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})^T \Delta\mathbf{x} + \sum_{j=1}^{l} \lambda_j \nabla_{\mathbf{x}} g_j(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})^T \Delta\mathbf{x} + \gamma^T \mathbf{A}\Delta\mathbf{x} = 0, \tag{20}$$

which leads to a contradiction. Therefore, there cannot exist a $\Delta\mathbf{x}$ that satisfies all constraints of the MIQP subproblems without a change in the $\mathbf{y}$-variables. As stated in Lemma 3, there always exists a solution to the MIQP subproblems as long as the lower bound is not equal to the upper bound. Solving the MIQP subproblem will, therefore, result in a new integer combination different from all previously obtained solutions. □

Note that, no assumptions were made in Theorem 2 regarding optimality of the MIQP subproblem. Therefore, the theorem is true for any solution that satisfies all constraints of the MIQP subproblem, optimal or not. Furthermore, Theorem 2 holds even if we make an arbitrary change to the objective function in the MIQP subproblems. An estimate of the Hessian in Q-OA will, therefore, be sufficient for Theorem 2 to hold. The next theorem summarizes the convergence properties.

Theorem 6. *Both L-OA and Q-OA will terminate after a finite number of iterations, either by verifying optimality of the best-found solution or proving that the MINLP problem is infeasible.*

Proof. From Lemma 1, it is clear that solving problem (OA-master) will either give a valid lower bound or prove infeasibility. Furthermore, the proof of Lemma 1 also shows that no feasible solution will be excluded from the search space. According to Theorem 2, both L-OA and Q-OA will find new integer combinations at each iteration as long as the gap between the upper and lower bound is not equal to zero. Since the linear constraints are assumed to give rise to a compact set, it is clear that there can only exist a finite number of different integer combinations, and thus, both methods must terminate after a finite number of iterations. □

Hence, we have proved that both proposed methods converge to a global optimal solution in a finite number of iterations. In the next section, we present a numerical comparison of the proposed methods and compare the results to the original OA method.

6 Computational results

In this section, we discuss our computational experiments and the obtained results. To compare the practical performance of the methods, we have implemented the original OA as well as L-OA and Q-OA. The main advantage of L-OA and Q-OA compared to the original OA, is the ability to handle highly

nonlinear MINLP problems more efficiently. L-OA is more conservative when choosing the trial solutions, and tries to stay close to the best found feasible solution, which should reduce the number of infeasible integer combinations obtained. In Q-OA we are also able to incorporate second order information when choosing new integer combinations. Hence, the new integer combination is chosen with information regarding the curvature around the current solution. From the test problems, we observed a significant reduction of the number of iterations with both L-OA and Q-OA compared to the original OA.

To test and compare the methods we have implemented them and applied them to convex MINLP problems obtained from MINLPlib2 (rev. 373, as of 2017-11-07)[1] [30]. This set was chosen since it contains a large variety of different test problems originating from both practical applications as well as theoretical test problems. As mentioned earlier both L-OA and Q-OA are intended for problems with high to medium degrees of non-linearity, and therefore, we used the following criteria for choosing the test problems

1. Classified as convex.
2. Having at least one discrete variable.
3. Having at least one continuous variable.
4. Satisfying the following inequality

$$\frac{n_{nonlin}}{n+m} > 0.5, \tag{21}$$

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 109 convex MINLP problems in MINLPlib2 (rev. 373, as of 2017-11-07) that satisfy the given criteria. These problems originate from several applications such as from process synthesis, facility layout problems, batch design with storage, portfolio optimization and MINLP test problems. The test instances have between 7 and 4530 variables and 0 to 1822 constraints. More details of the test instances are provided in the supplemental material.

Next, we describe some details regarding the implementation of the methods and the computational results are presented in section 6.2 and 6.3.

6.1 Implementation details

The implementation of the methods compared here was made in MATLAB using Gurobi 7.5.1[28] as subsolver for the MILP/MIQP subproblems and IPOPT 3.12.7 [31] for the NLP subproblems. Furthermore, we use some functionality from OPTI Toolbox [32] to read the test problems.

Both L-OA and Q-OA require a feasible starting solution, and to obtain such a solution we start by performing a few original OA iterations. Once a feasible

[1] `http://www.gamsworld.org/minlp/minlplib2/html/index.html`

solution is obtained, we switch to either L-OA or Q-OA. The level parameter α was set to 0.5 with both methods in the comparison.

According to Theorem 2, it is not necessary to find the optimal solution for the MIQP subproblems, and any feasible solution for these problems is sufficient for guaranteeing that both L-OA and Q-OA converge to the global optimum. This is an important property, since solvers such as Gurobi or CPLEX are often able to quickly find several feasible solutions, and quite often the majority of the solution time is spent proving optimality. Here we use a strategy of stopping the solver once a certain number of feasible solution have been found, and specifically, we stop after 10 solutions have been found. This is simply done by setting the SolutionLimit parameter to 10. By this approach, we ensure we obtain a good solution to the MIQP problem, while significantly reducing the total solution time. For the MIQP subproblems, we always have a feasible solution available, the solution to the MILP subproblem (OA-master), and providing this as a starting solution to Gurobi also improved the performance. For the MILP subproblems, we used the default settings in Gurobi, and we also used the default settings for IPOPT.

The NLP subproblem (NLP-I) is always convex for these test problems. However, for some specific test problems we encountered some difficulties where the solver failed to find the optimal solution. Such difficulties could, for example, be caused by a specific integer combination not satisfying the constraint qualifications. These issues were not frequent and they only occurred for a few test problem in the entire MINLPLib2. To deal with such issues we chose a simple approach; if the NLP subproblem (NLP-I) is feasible but the NLP solver fails, we generate cutting planes for all violated constraints at the solution given by the MILP subproblem (OA-master) according to eq. (1). These cuts will exclude the current solution to subproblem (OA-master) from the search space [33], and thus prevent cycling. Adding these cuts is equivalent to performing an iteration with the ECP method. From the convergence properties of the ECP method, we know that adding these cuts will eventually result in a new integer combination or verify optimality of an obtained solution.

Since the problems we consider are all convex, the Hessian of the Lagrangean is always positive semidefinite. However, due to numerical accuracy we did encounter a few cases where the Hessian was not strictly positive semidefinite, *i.e.,* the smallest eigenvalue was not positive but in the range of -10^{-9}. To make sure that the MIQP subproblems are convex, we slightly modify the diagonal elements of the Hessian. For each row i of the Hessian which contains a nonzero element, we modify the diagonal by

$$\nabla^2_{\mathbf{x},\mathbf{y}}\mathcal{L}(i,i) := \nabla^2_{\mathbf{x},\mathbf{y}}\mathcal{L}(i,i) + |\lambda_{min}|, \tag{22}$$

where λ_{min} is chosen as the smallest eigenvalue of the Hessian. This modification guarantees that all eigenvalues are positive [34], and thus, ensures convexity of the MIQP subproblem. The modification of the Hessian is only done in case one of the eigenvalues are negative.

As termination criteria, we used both an absolute optimality tolerance ϵ and

a relative optimality tolerance ϵ_{rel}. The search is, thus terminated if either

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

are satisfied. Here LB denotes the current lower bound. These can be considered as the standard termination criteria for MINLP problems.

All tests were performed on an Intel Core i7 2.93GHz CPU desktop with 16GB of RAM running Windows 7, and as termination criteria, we set the tolerances $\epsilon = 10^{-5}$ and $\epsilon_{rel} = 10^{-3}$ and a time limit of $900s$.

6.2 Illustrative examples

In this section we present more detailed results of two particular instances of the selected test set. These instances were chosen such that they could exemplify the results shown in the following section. The selected instances are `cvxnonsep_nsig40` [2] and `ibs2` [3]. The first instances were proposed by Kronqvist et al. [35], it contains 20 integer variables and 20 continuous variables a linear objective and a signomial constraints. This seemingly simple problem is designed to be challenging for methods such as OA and ECP due to a highly nonlinear constraint. The second instance has 1500 binary variables, 1510 continuous variables, a linear objective, and 1821 constraints, of which 10 are nonlinear including square and logarithm operators. This problem represents a particular challenge for the OA method given its combinatorial complexity and the fact that most of its variables, discrete and continuous, are involved in a nonlinear fashion in the constraints.

To illustrate how the methods differ for these problems, we show the upper and lower bounds obtained by each method. Figures 5 and 6 show the evolution of the bounds as a function of time for problems `cvxnonsep_nsig40` and `ibs2`, respectively. From the figures, it can be observed that Q-OA is able to improve the upper bound more quickly than the other methods. This is usually the case and is explained by that fact that Q-OA utilizes more information when choosing the integer combinations than the other methods. Especially for the instance `ibs2`, there was a clear advantage of incorporating information of the second order derivatives, and Q-OA clearly performs better than L-OA.

From the results presented in the bounds profiles and in the Table 1, we notice how the inclusion of level regularization can improve the performance of the OA method while solving convex MINLPs. For the instance `cvxnonsep_nsig40` we notice a reduction in time of 59% and 66% using the L-OA and the Q-OA method, respectively. Although the new methods require the solution of an MIQP subproblem in each iteration, the extra time invested in finding the next integer combination is compensated with a reduction in both iterations and time.

For the instance `ibs2`, OA is unable to close the optimality gap under 0.1% within the time limit of 900 seconds even though it performs 431 iterations.

[2] `http://www.gamsworld.org/minlp/minlplib2/html/cvxnonsep_nsig40.html`
[3] `http://www.gamsworld.org/minlp/minlplib2/html/ibs2.html`

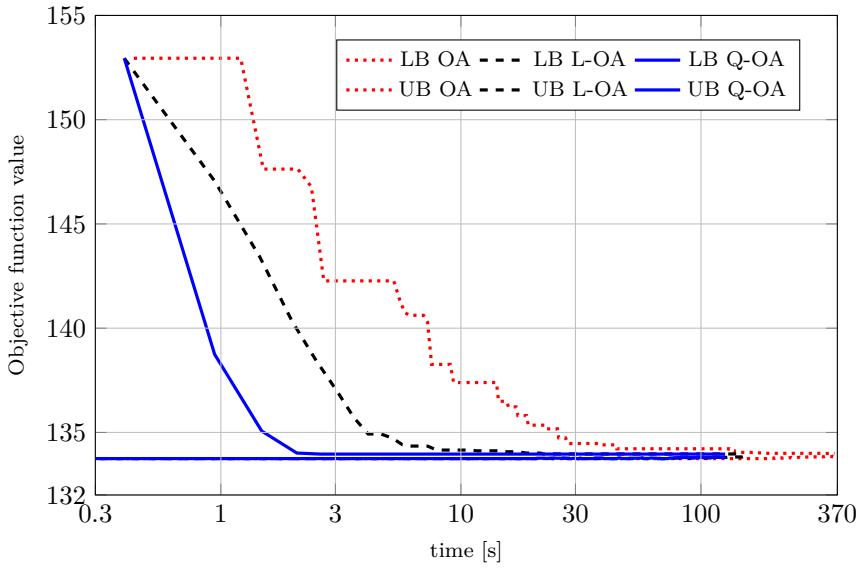

Figure 5: Bound profiles for instance `cvxnonsep_nsig40` against time. The figure shows the upper bound (UB) and lower bound (LB) obtained by the OA, L-OA, and Q-OA methods.

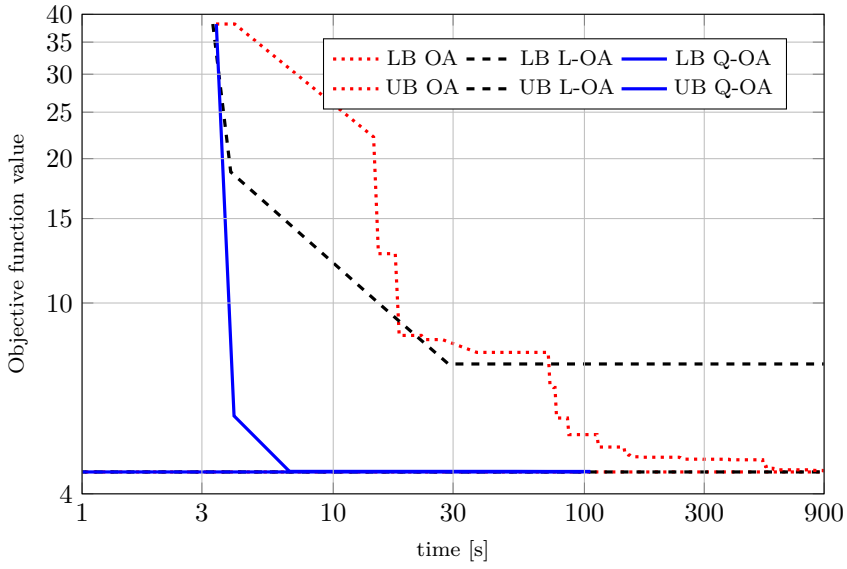

Figure 6: Bound profiles for instance `ibs2` against time. The figure shows the upper bound (UB) and lower bound (LB) obtained by the OA, L-OA, and Q-OA methods.

Table 1: Detailed results of the illustrative examples while solving them with the OA, L-OA, and Q-OA methods

Instance	Solution method	Time [s]	Iterations	NLP time [s]	MILP time [s]	MIQP time [s]	Infeasible NLPs	Optimality Gap
cvxnonsep_nsig40	OA	360.86	663	31.69	328.13	0	1	0.00098
	L-OA	147.87	201	9.84	55.35	82.37	0	0.000999
	Q-OA	121.55	144	6.65	38.68	75.99	0	0.000913
ibs2	OA	900*	431	152.86	747.14	0	6	0.005368
	L-OA	900*	41	40.16	17.98	841.86	6	0.1093
	Q-OA	103.89	16	48.60	6.469	48.68	2	0.000997

*Time limit.

When solving the problem with L-OA the upper bound initially diminishes faster in terms of both time and iterations compared to OA, the MIQP subproblems become hard to solve resulting in only 41 iterations before hitting the time limit. When utilizing second order information with the Q-OA method, the problem is solved within an optimality gap of 0.1% in 104 seconds and just after 16 iterations while only encountering 2 infeasible NLP subproblems. Note that the lower bounds found for both illustrative examples are close to the optimal value for all methods, given that they were obtained by solving the continuous relaxation of the MINLP problems.

6.3 Numerical results

Having observed the improvement in performance of the proposed methods compared to OA in the illustrative examples, we considered the whole test set defined at the beginning of this section. In order to compare the performance of the methods, we have used performance profiles [36] both in terms of solution time and iterations in Figures 7 and 8, respectively. The profiles show the number of problems solved against the respective performance ratio threshold τ. A data point at each plot represents the number of instances that each method solved within a factor τ of the best solver.

Figure 7 shows how the Q-OA method is superior to both L-OA and OA in the selected test set in terms of solution time. The figure shows that Q-OA solves most instances to the desired optimality gap, and it solves the problems in the least amount of time. L-OA has initially the worst performance of the 3 methods for $\tau \leq 3$, but in the end, the performance is similar to that of OA without reaching the number of solved instances by Q-OA. It is also worth mentioning, that all the instances that remained unsolved with Q-OA are also unsolved with both OA and L-OA. Q-OA is thus able to solve all the problems solved with the other methods and some additional problems.

The performance profiles in terms of iterations in Figure 8 show a clear advantage of Q-OA compared to the other 2 methods. Considering iterations the L-OA method performs better than OA whenever the iterations factor $\tau_{iter} \geq 1.5$.

Given that the performance profiles show the results without distinguishing the individual instances, we include Table 2, which shows a direct comparison

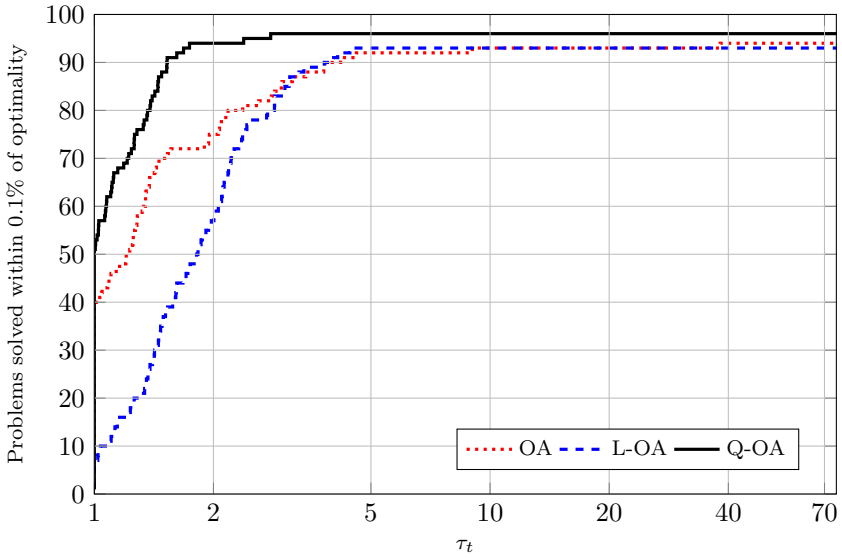

Figure 7: Time performance profiles for test problems.

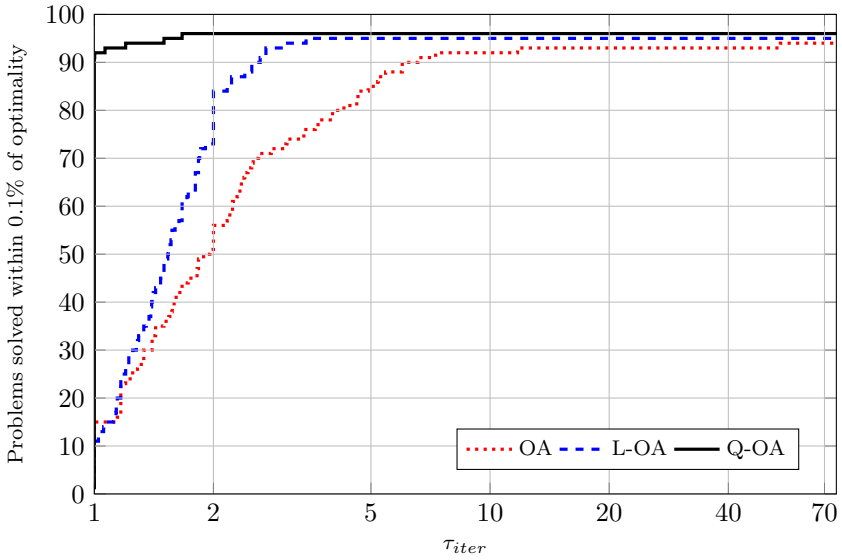

Figure 8: Iterations performance profiles for test problems.

Table 2: Number of instances solved and comparison in solution time and iterations of OA, L-OA, and Q-OA.

Method	Instances solved	Less time than OA	Fewer iterations than OA	Less time than L-OA	Fewer iterations than L-OA	Less time than Q-OA	Fewer iterations than Q-OA
OA	94 / 109	-	-	61 / 94	23 / 94	38 / 94	3 / 94
L-OA	95 / 109	34 / 95	57 / 95	-	-	9 / 94	2 / 94
Q-OA	96 / 109	57 / 96	80 / 96	86 / 96	84 / 96	-	-

of the methods in terms of solution time and iterations. Note that Q-OA is able to solve 1 and 2 instances more than L-OA and OA, respectively.

None of the methods was able to find a solution within 0.1% of optimality gap for 13 instances in the test set. When comparing the proposed methods to OA, we see that L-OA is able to reduce the solution time in 36% of the instances and the iterations in 60%, while Q-OA reduced the time in 59% of the instances and the iterations in 83%. From the results, it was also noticed that the benefits of Q-OA are more apparent for the more challenging instances. Comparing the two proposed methods we see that Q-OA solves 90% of the instances in less time and 87.5% of the instances is fewer iterations than L-OA. Detailed solution information for all instances and methods can be found in the supplemental material.

An interesting result is that the proposed methods significantly decreased the number of infeasible NLP subproblems found while solving the selected problems. Using OA we obtained 877 infeasible NLP subproblems while using L-OA and Q-OA we obtained 259 and 257, respectively. This can be explained by the fact that the integer combinations are chosen closer in the search space to the best feasible solution, and information about the curvature is utilized with the proposed methods. Choosing an integer combination close to a feasible solution also results in trial solutions close to the feasible region, which resulted in fewer infeasible trial solutions.

The solutions reported here were obtained using the level parameter $\alpha = 0.5$. Changing the value of α affects the performance of the proposed methods for the particular instances. We performed several tests varying the value of α, which resulted in significant changes for individual instances but insignificant when considering the whole test set.

7 Conclusions and future work

We have presented two new methods for solving convex MINLP problems, based on a regularization technique and a second order approximation of the Lagrangean. We have proven that both methods converge to the global optimal solution in a finite number of iterations, and shown that the proofs hold, even if the MIQP subproblem is only solved approximately. Both methods are mainly intended for problems with moderate to high degrees of nonlinearity, and for such problems, both methods performed better than the original OA. The new method called Q-OA required significantly fewer iteration than the original OA

and there was also a clear advantage in the solution time. The advantage is due to the fact that more information is utilized when choosing the integer combinations. The method L-OA uses a regularization technique which we showed is equivalent to using a trust region. The regularization prevents large jumps between iterations and tries to keep the trial solutions close to the feasible region, and for the test problems, it gave an advantage over the original OA. For the test problems, Q-OA performed better than the other methods, both with respect to the number of iteration and time and furthermore, we were able to solve a greater percentage of problems within the time limit with Q-OA.

As future work we plan to implement the methods in a more efficient and flexible framework, *e.g.,* within an MINLP solver like DICOPT[37] or as part of a Toolkit in an optimization modeling software such as Pyomo or JuliaOpt. It could also be worth to investigate a dynamic update of the level parameter α. For example, it could be possible to adjust the parameter based on the current optimality gap.

Acknowledgements

Jan Kronqvist is grateful for the grants given by Walter Ahlström foundation, Svenska tekniska vetenskapsakademien i Finland, Tekniikan edistämissäätiö, TFIF and Waldemar von Frenckells stiftelse, which made the research visit at Carnegie Mellon University possible. David E. Bernal and Ignacio E. Grossmann would like to thank the Center Advanced Process Decision Making (CAPD) for its financial support.

References

[1] C. A. Floudas, "Deterministic Global Optimization, vol. 37 of Nonconvex Optimization and its Applications," 2000.

[2] L. T. Biegler and I. E. Grossmann, "Retrospective on optimization," *Computers & Chemical Engineering*, vol. 28, no. 8, pp. 1169–1192, 2004.

[3] J. Lee and S. Leyffer, eds., *Mixed Integer Nonlinear Programming*, vol. 154. Springer Science & Business Media, 2011.

[4] M. A. Duran and I. E. Grossmann, "An outer-approximation algorithm for a class of mixed-integer nonlinear programs," *Mathematical Programming*, vol. 36, no. 3, pp. 307–339, 1986.

[5] T. Westerlund and F. Petterson, "An extended cutting plane method for solving convex MINLP problems," *Computers & Chemical Engineering*, vol. 19, pp. S131–S136, 1995.

[6] J. Kronqvist, A. Lundell, and T. Westerlund, "The extended supporting hyperplane algorithm for convex mixed-integer nonlinear programming," *Journal of Global Optimization*, vol. 64, no. 2, pp. 249–272, 2016.

[7] A. M. Geoffrion, "Generalized Benders decomposition," *Journal of Optimization Theory and Applications*, vol. 10, no. 4, pp. 237–260, 1972.

[8] R. J. Dakin, "A tree-search algorithm for mixed integer programming problems," *The Computer Journal*, vol. 8, no. 3, pp. 250–255, 1965.

[9] P. Bonami, L. T. Biegler, A. R. Conn, G. Cornuéjols, I. E. Grossmann, C. D. Laird, J. Lee, A. Lodi, F. Margot, N. Sawaya, and A. Wächter, "An algorithmic framework for convex mixed integer nonlinear programs," *Discrete Optimization*, vol. 5, no. 2, pp. 186–204, 2008.

[10] P. Belotti, C. Kirches, S. Leyffer, J. Linderoth, J. Luedtke, and A. Mahajan, "Mixed-integer nonlinear optimization," *Acta Numerica*, vol. 22, pp. 1–131, 2013.

[11] F. Trespalacios and I. E. Grossmann, "Review of mixed-integer nonlinear and generalized disjunctive programming methods," vol. 86, no. 7, pp. 991–1012, 2014.

[12] J. E. Kelley, Jr, "The cutting-plane method for solving convex programs," *Journal of the Society for Industrial & Applied Mathematics*, vol. 8, no. 4, pp. 703–712, 1960.

[13] Y. Nesterov, *Introductory Lectures on Convex Optimization: A Basic Course*, vol. 87. Springer, 2004.

[14] D. den Hertog, J. Kaliski, C. Roos, and T. Terlaky, "A logarithmic barrier cutting plane method for convex programming," *Annals of Operations Research*, vol. 58, no. 2, pp. 67–98, 1995.

[15] A. Bagirov, N. Karmitsa, and M. M. Mäkelä, *Introduction to Nonsmooth Optimization: Theory, Practice and Software*. Springer, 2014.

[16] S. Zaourar and J. Malick, "Quadratic stabilization of Benders decomposition." working paper or preprint, 2014.

[17] W. de Oliveira, "Regularized optimization methods for convex MINLP problems," *TOP*, vol. 24, no. 3, pp. 665–692, 2016.

[18] C. Lemaréchal, A. Nemirovskii, and Y. Nesterov, "New variants of bundle methods," *Mathematical Programming*, vol. 69, no. 1-3, pp. 111–147, 1995.

[19] K. C. Kiwiel, "Proximal level bundle methods for convex nondifferentiable optimization, saddle-point problems and variational inequalities," *Mathematical Programming*, vol. 69, no. 1-3, pp. 89–109, 1995.

[20] R. Fletcher and S. Leyffer, "Solving mixed integer nonlinear programs by outer approximation," *Mathematical Programming*, vol. 66, no. 1, pp. 327–349, 1994.

[21] GAMS World, "Mixed-integer nonlinear programming library," 2017. [Online; accessed 13-November-2017].

[22] M. Slater *et al.*, "Lagrange multipliers revisited," tech. rep., Cowles Foundation for Research in Economics, Yale University, 1959.

[23] Z. Wei and M. M. Ali, "Outer Approximation Algorithm for One Class of Convex Mixed-Integer Nonlinear Programming Problems with Partial Differentiability," *Journal of Optimization Theory and Applications*, vol. 167, no. 2, pp. 644–652, 2015.

[24] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[25] J. Viswanathan and I. E. Grossmann, "A combined penalty function and outer-approximation method for MINLP optimization," *Computers & Chemical Engineering*, vol. 14, no. 7, pp. 769–782, 1990.

[26] P. Bonami, G. Cornuéjols, A. Lodi, and F. Margot, "A feasibility pump for mixed integer nonlinear programs," *Mathematical Programming*, vol. 119, no. 2, pp. 331–352, 2009.

[27] P. Wolfe, "A duality theorem for non-linear programming," *Quarterly of applied mathematics*, vol. 19, no. 3, pp. 239–244, 1961.

[28] I. Gurobi Optimization, "Gurobi optimizer reference manual," 2016.

[29] IBM Corp. and IBM, "V12.6: User's Manual for CPLEX," *International Business Machines Corporation*, vol. 12, no. 1, p. 481, 2009.

[30] GAMSWorld, "Mixed-integer nonlinear programming library," 2016. [Online; accessed 24-November-2016].

[31] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, no. 1, pp. 25–57, 2006.

[32] J. Currie and D. I. Wilson, "OPTI: Lowering the Barrier Between Open Source Optimizers and the Industrial MATLAB User," in *Foundations of Computer-Aided Process Operations* (N. Sahinidis and J. Pinto, eds.), (Savannah, Georgia, USA), 8–11 January 2012.

[33] T. Westerlund and R. Pörn, "Solving pseudo-convex mixed integer optimization problems by cutting plane techniques," *Optimization and Engineering*, vol. 3, no. 3, pp. 253–280, 2002.

[34] S. A. Gershgorin, "Uber die Abgrenzung der Eigenwerte einer Matrix," *Bulletin de l'Académie des Sciences de l'URSS. Classe des sciences mathématiques et na*, no. 6, pp. 749–754, 1931.

[35] J. Kronqvist, A. Lundell, and T. Westerlund, "Reformulations for utilizing separability when solving convex MINLP problems," *Submitted for publication*, 2017.

[36] E. D. Dolan and J. J. Moré, "Benchmarking optimization software with performance profiles," *Mathematical Programming, Series B*, vol. 91, no. 2, pp. 201–213, 2002.

[37] I. E. Grossmann, J. Viswanathan, A. Vecchietti, R. Raman, and E. Kalvelagen, "GAMS/DICOPT: A Discrete Continuous Optimization Package," 2002.

A Test set

The main properties of the test problems are shown in Table 3, showing the number of integer variables m^{int}, binary variables m^{bin}, continuous variables n, and total variables $n+m$. The table also shows the number of constraints, the number of nonlinear functions, *i.e.,* nonlinear objective function and constraint functions, and the ratio between number of variables present in a nonlinear term and the total of variables $\frac{n_{nonlin}}{n+m}$.

Table 3: List of instances in convex MINLP test set.

Instance	m^{int}	m^{bin}	n	$n+m$	ncons	nnlfunc	$\frac{n_{nonlin}}{n+m}$
cvxnonsep_normcon40	20	0	20	40	1	1	1
cvxnonsep_nsig40	20	0	20	40	1	1	1
cvxnonsep_pcon40	20	0	20	40	1	1	1
cvxnonsep_psig40	20	0	20	40	0	1	1
cvxnonsep_normcon30	15	0	15	30	1	1	1
cvxnonsep_nsig30	15	0	15	30	1	1	1
cvxnonsep_pcon30	15	0	15	30	1	1	1
cvxnonsep_psig30	15	0	15	30	0	1	1
cvxnonsep_normcon20	10	0	10	20	1	1	1
cvxnonsep_nsig20	10	0	10	20	1	1	1
cvxnonsep_pcon20	10	0	10	20	1	1	1
cvxnonsep_psig20	10	0	10	20	0	1	1
du-opt	13	0	7	20	9	1	1
du-opt5	13	0	7	20	9	1	1
ex1223b	0	4	3	7	9	5	1
ibs2	0	1500	1510	3010	1821	10	0.996678
squfl030-150	0	30	4500	4530	4650	1	0.993377
squfl020-150	0	20	3000	3020	3150	1	0.993377
smallinvSNPr1b010-011	100	0	1	101	4	1	0.990099
smallinvSNPr1b020-022	100	0	1	101	4	1	0.990099
smallinvSNPr1b050-055	100	0	1	101	4	1	0.990099
smallinvSNPr1b100-110	100	0	1	101	4	1	0.990099

smallinvSNPr1b150-165	100	0	1	101	4	1	0.990099
smallinvSNPr1b200-220	100	0	1	101	4	1	0.990099
smallinvSNPr2b010-011	100	0	1	101	4	1	0.990099
smallinvSNPr2b020-022	100	0	1	101	4	1	0.990099
smallinvSNPr2b050-055	100	0	1	101	4	1	0.990099
smallinvSNPr2b100-110	100	0	1	101	4	1	0.990099
smallinvSNPr2b150-165	100	0	1	101	4	1	0.990099
smallinvSNPr2b200-220	100	0	1	101	4	1	0.990099
smallinvSNPr3b010-011	100	0	1	101	4	1	0.990099
smallinvSNPr3b020-022	100	0	1	101	4	1	0.990099
smallinvSNPr3b050-055	100	0	1	101	4	1	0.990099
smallinvSNPr3b100-110	100	0	1	101	4	1	0.990099
smallinvSNPr3b150-165	100	0	1	101	4	1	0.990099
smallinvSNPr3b200-220	100	0	1	101	4	1	0.990099
smallinvSNPr4b010-011	100	0	1	101	4	1	0.990099
smallinvSNPr4b020-022	100	0	1	101	4	1	0.990099
smallinvSNPr4b050-055	100	0	1	101	4	1	0.990099
smallinvSNPr4b100-110	100	0	1	101	4	1	0.990099
smallinvSNPr4b150-165	100	0	1	101	4	1	0.990099
smallinvSNPr4b200-220	100	0	1	101	4	1	0.990099
smallinvSNPr5b010-011	100	0	1	101	4	1	0.990099
smallinvSNPr5b020-022	100	0	1	101	4	1	0.990099
smallinvSNPr5b050-055	100	0	1	101	4	1	0.990099
smallinvSNPr5b100-110	100	0	1	101	4	1	0.990099
smallinvSNPr5b150-165	100	0	1	101	4	1	0.990099
smallinvSNPr5b200-220	100	0	1	101	4	1	0.990099
squfl030-100	0	30	3000	3030	3100	1	0.990099
squfl040-080	0	40	3200	3240	3280	1	0.987654
squfl015-080	0	15	1200	1215	1280	1	0.987654
squfl010-080	0	10	800	810	880	1	0.987654
squfl015-060	0	15	900	915	960	1	0.983607
squfl020-050	0	20	1000	1020	1050	1	0.980392
squfl025-040	0	25	1000	1025	1040	1	0.97561
squfl020-040	0	20	800	820	840	1	0.97561
squfl010-040	0	10	400	410	440	1	0.97561
smallinvDAXr1b010-011	30	0	1	31	4	1	0.967742
smallinvDAXr1b020-022	30	0	1	31	4	1	0.967742
smallinvDAXr1b050-055	30	0	1	31	4	1	0.967742
smallinvDAXr1b100-110	30	0	1	31	4	1	0.967742
smallinvDAXr1b150-165	30	0	1	31	4	1	0.967742
smallinvDAXr1b200-220	30	0	1	31	4	1	0.967742
smallinvDAXr2b010-011	30	0	1	31	4	1	0.967742
smallinvDAXr2b020-022	30	0	1	31	4	1	0.967742
smallinvDAXr2b050-055	30	0	1	31	4	1	0.967742
smallinvDAXr2b100-110	30	0	1	31	4	1	0.967742
smallinvDAXr2b150-165	30	0	1	31	4	1	0.967742

smallinvDAXr2b200-220	30	0	1	31	4	1	0.967742
smallinvDAXr3b010-011	30	0	1	31	4	1	0.967742
smallinvDAXr3b020-022	30	0	1	31	4	1	0.967742
smallinvDAXr3b050-055	30	0	1	31	4	1	0.967742
smallinvDAXr3b100-110	30	0	1	31	4	1	0.967742
smallinvDAXr3b150-165	30	0	1	31	4	1	0.967742
smallinvDAXr3b200-220	30	0	1	31	4	1	0.967742
smallinvDAXr4b010-011	30	0	1	31	4	1	0.967742
smallinvDAXr4b020-022	30	0	1	31	4	1	0.967742
smallinvDAXr4b050-055	30	0	1	31	4	1	0.967742
smallinvDAXr4b100-110	30	0	1	31	4	1	0.967742
smallinvDAXr4b150-165	30	0	1	31	4	1	0.967742
smallinvDAXr4b200-220	30	0	1	31	4	1	0.967742
smallinvDAXr5b010-011	30	0	1	31	4	1	0.967742
smallinvDAXr5b020-022	30	0	1	31	4	1	0.967742
smallinvDAXr5b050-055	30	0	1	31	4	1	0.967742
smallinvDAXr5b100-110	30	0	1	31	4	1	0.967742
smallinvDAXr5b150-165	30	0	1	31	4	1	0.967742
smallinvDAXr5b200-220	30	0	1	31	4	1	0.967742
squfl025-030	0	25	750	775	780	1	0.967742
squfl025-025	0	25	625	650	650	1	0.961538
squfl010-025	0	10	250	260	275	1	0.961538
fac2	0	12	54	66	33	1	0.818182
fac3	0	12	54	66	33	1	0.818182
fac1	0	6	16	22	18	1	0.727273
ex1223	0	4	7	11	13	5	0.636364
st_e14	0	4	7	11	13	5	0.636364
batchdes	0	9	10	19	19	2	0.526316
cvxnonsep_pcon20r	10	0	29	39	20	19	0.512821
cvxnonsep_pcon30r	15	0	44	59	30	29	0.508475
cvxnonsep_pcon40r	20	0	59	79	40	39	0.506329
cvxnonsep_psig40r	20	0	62	82	42	41	0.5
cvxnonsep_normcon40r	20	0	60	80	41	40	0.5
cvxnonsep_nsig40r	20	0	60	80	41	40	0.5
cvxnonsep_psig30r	15	0	47	62	32	31	0.5
cvxnonsep_normcon30r	15	0	45	60	31	30	0.5
cvxnonsep_nsig30r	15	0	45	60	31	30	0.5
cvxnonsep_psig20r	10	0	32	42	22	21	0.5
cvxnonsep_normcon20r	10	0	30	40	21	20	0.5
cvxnonsep_nsig20r	10	0	30	40	21	20	0.5
st_miqp4	3	0	3	6	4	1	0.5

B Detailed results

Details regarding the computational results are given in Table 4, showing the number of iterations, time and final gap for each test problem and method. The sign $*$ indicates that the time limit was exceeded. The gap reported is calculated as $|LB - UB|/|LB|$ where LB and UB are the lower and upper bounds, respectively.

Table 4: Detailed results

	OA			L-OA			Q-OA		
Instance	Time [s]	Iterations	Gap	Time [s]	Iterations	Gap	Time [s]	Iterations	Gap
cvxnonsep_normcon40	900*	1856*	0.0123*	900*	549*	0.0076*	900*	535*	0.0075*
cvxnonsep_nsig40	360.863	663	0.0010	147.874	201	0.0010	121.554	144	0.0009
cvxnonsep_pcon40	119.204	290	0.0010	112.095	158	0.0009	108.194	148	0.0006
cvxnonsep_psig40	322.982	629	0.0010	168.723	210	0.0009	165.35	205	0.0009
cvxnonsep_normcon30	900*	1134*	0.0016*	538.453	455	0.0009	574.245	456	0.0006
cvxnonsep_nsig30	381.401	915	0.0010	42.286	78	0.0009	48.464	83	0.0009
cvxnonsep_pcon30	52.22	156	0.0010	41.269	72	0.0009	37.914	60	0.0008
cvxnonsep_psig30	266.827	694	0.0009	98.742	160	0.0007	89.545	141	0.0009
cvxnonsep_normcon20	104.701	311	0.0007	55.874	114	0.0010	49.974	102	0.0009
cvxnonsep_nsig20	35.909	119	0.0009	26.007	50	0.0010	16.489	32	0.0009
cvxnonsep_pcon20	19.183	63	0.0010	17.829	35	0.0009	13.166	25	0.0008
cvxnonsep_psig20	113.183	379	0.0008	9.25	19	0.0010	2.967	7	0.0008
du-opt	1.223	5	0.0005	4.066	9	0.0009	0.919	3	0.0000
du-opt5	1.203	5	0.0002	2.541	6	0.0008	0.89	3	0.0001
ex1223b	1.53	5	0.0000	2.225	5	0.0000	2.116	5	0.0000
squfl030-150	900*	88*	3.8776*	900*	9*	1.7116*	900*	13*	2.0196*
squfl020-150	900*	98*	2.7303*	900*	19*	1.4643*	900*	42*	1.4822*
smallinvSNPr1b010-011	1.894	6	0.0000	4.122	8	0.0000	2.35	5	0.0000
smallinvSNPr1b020-022	2.208	7	0.0000	4.63	9	0.0000	3.063	6	0.0000
smallinvSNPr1b050-055	3.18	10	0.0000	6.182	11	0.0000	4.249	8	0.0000
smallinvSNPr1b100-110	3.534	10	0.0000	7.272	13	0.0000	3.627	7	0.0008
smallinvSNPr1b150-165	3.412	11	0.0007	5.254	10	0.0003	2.535	5	0.0003
smallinvSNPr1b200-220	3.879	12	0.0006	5.994	12	0.0008	2.483	5	0.0008
smallinvSNPr2b010-011	1.194	4	0.0000	2.965	6	0.0000	1.819	4	0.0000
smallinvSNPr2b020-022	2.09	7	0.0000	4.63	9	0.0000	3.026	6	0.0000
smallinvSNPr2b050-055	3.051	10	0.0000	5.514	10	0.0000	2.483	5	0.0000
smallinvSNPr2b100-110	3.858	12	0.0003	5.91	11	0.0009	4.864	9	0.0007
smallinvSNPr2b150-165	3.212	10	0.0004	5.389	10	0.0000	2.532	5	0.0004
smallinvSNPr2b200-220	4.059	13	0.0004	6.539	12	0.0000	4.364	8	0.0000
smallinvSNPr3b010-011	2.301	7	0.0000	3.181	6	0.0000	2.73	5	0.0000
smallinvSNPr3b020-022	2.026	6	0.0000	3.673	7	0.0000	3.095	6	0.0000
smallinvSNPr3b050-055	3.084	10	0.0000	5.639	10	0.0000	4.318	8	0.0000
smallinvSNPr3b100-110	3.213	10	0.0006	4.998	9	0.0006	2.375	5	0.0006
smallinvSNPr3b150-165	3.412	11	0.0000	5.286	10	0.0009	3.156	6	0.0007
smallinvSNPr3b200-220	4.238	13	0.0000	6.489	12	0.0008	3.073	6	0.0001
smallinvSNPr4b010-011	7.176	20	0.0000	9.898	16	0.0000	9.05	14	0.0000
smallinvSNPr4b020-022	2.192	7	0.0000	3.523	7	0.0000	3.117	6	0.0000
smallinvSNPr4b050-055	3.477	11	0.0000	5.572	10	0.0003	3.733	7	0.0001
smallinvSNPr4b100-110	3.686	12	0.0000	6.667	12	0.0000	4.724	9	0.0000

smallinvSNPr4b150-165	4.373	14	0.0000	5.88	11	0.0002	3.64	7	0.0000
smallinvSNPr4b200-220	3.351	11	0.0000	5.726	11	0.0000	3.077	6	0.0000
smallinvSNPr5b010-011	4.618	14	0.0000	9.93	16	0.0000	7.46	12	0.0000
smallinvSNPr5b020-022	2.841	9	0.0000	3.579	7	0.0000	3.15	6	0.0000
smallinvSNPr5b050-055	3.479	11	0.0000	5.51	10	0.0000	2.443	5	0.0000
smallinvSNPr5b100-110	2.751	8	0.0000	6.007	11	0.0000	3.657	7	0.0000
smallinvSNPr5b150-165	4.421	14	0.0005	5.848	11	0.0008	3.119	6	0.0005
smallinvSNPr5b200-220	3.785	12	0.0000	5.276	10	0.0000	3.023	6	0.0000
squfl030-100	900*	113*	3.2451*	900*	11*	1.9290*	900*	48*	1.9899*
squfl040-080	900*	123*	3.4900*	900*	10*	1.8806*	900*	46*	1.8806*
squfl015-080	900*	152*	1.1302*	900*	71*	1.2469*	900*	75*	1.2660*
squfl010-080	900*	181*	0.1943*	900*	121*	0.2657*	900*	132*	0.2509*
squfl015-060	900*	200*	0.4635*	900*	130*	0.4371*	900*	110*	0.8024*
squfl020-050	900*	190*	1.1339*	900*	88*	1.2583*	900*	99*	1.3194*
squfl025-040	900*	199*	1.1967*	900*	60*	1.5676*	900*	98*	1.5676*
squfl020-040	900*	231*	0.3661*	900*	112*	0.5885*	900*	114*	0.6238*
squfl010-040	238.235	159	0.0000	205.645	104	0.0001	206.827	101	0.0001
smallinvDAXr1b010-011	11.335	35	0.0008	14.291	25	0.0001	12.691	22	0.0008
smallinvDAXr1b020-022	12.043	38	0.0005	14.295	25	0.0005	10.002	17	0.0005
smallinvDAXr1b050-055	14.076	43	0.0006	14.803	25	0.0005	10.445	18	0.0005
smallinvDAXr1b100-110	16.173	51	0.0008	12.489	22	0.0009	6.212	11	0.0009
smallinvDAXr1b150-165	14.641	46	0.0009	14.059	24	0.0009	3.493	7	0.0009
smallinvDAXr1b200-220	14.787	47	0.0009	11.347	20	0.0008	4.671	9	0.0007
smallinvDAXr2b010-011	12.981	40	0.0010	14.629	25	0.0001	12.975	22	0.0008
smallinvDAXr2b020-022	12.396	39	0.0005	12.641	22	0.0004	9.675	17	0.0005
smallinvDAXr2b050-055	13.484	42	0.0008	12.23	22	0.0009	10.662	18	0.0005
smallinvDAXr2b100-110	13.918	44	0.0010	12.966	22	0.0008	5.869	11	0.0010
smallinvDAXr2b150-165	16.614	51	0.0010	10.966	19	0.0010	3.617	7	0.0009
smallinvDAXr2b200-220	16.22	49	0.0009	11.232	20	0.0006	4.741	9	0.0007
smallinvDAXr3b010-011	12.487	38	0.0010	18.347	31	0.0007	12.795	22	0.0008
smallinvDAXr3b020-022	12.574	38	0.0005	13.228	23	0.0009	9.789	17	0.0005
smallinvDAXr3b050-055	14.313	44	0.0006	14.523	25	0.0008	10.837	18	0.0005
smallinvDAXr3b100-110	15.528	47	0.0009	13.817	24	0.0006	7.497	13	0.0010
smallinvDAXr3b150-165	15.892	48	0.0010	11.882	21	0.0009	4.153	8	0.0009
smallinvDAXr3b200-220	16.602	52	0.0009	11.495	20	0.0006	5.817	10	0.0009
smallinvDAXr4b010-011	11.106	35	0.0007	12.699	22	0.0007	13.449	22	0.0008
smallinvDAXr4b020-022	13.016	40	0.0005	14.626	25	0.0006	10.425	17	0.0005
smallinvDAXr4b050-055	15.155	45	0.0009	13.019	23	0.0006	10.559	18	0.0005
smallinvDAXr4b100-110	14.402	44	0.0009	11.205	20	0.0009	7.405	13	0.0010
smallinvDAXr4b150-165	15.547	48	0.0010	11.21	20	0.0009	4.091	8	0.0009
smallinvDAXr4b200-220	14.571	44	0.0008	11.565	21	0.0009	6.78	11	0.0009
smallinvDAXr5b010-011	12.773	40	0.0006	12.884	22	0.0007	14.052	22	0.0008

smallinvDAXr5b020-022	11.054	34	0.0005	10.71	19	0.0005	10.791	18	0.0005
smallinvDAXr5b050-055	13.882	42	0.0008	13.168	23	0.0010	9.071	15	0.0008
smallinvDAXr5b100-110	14.225	44	0.0009	11.124	20	0.0007	7.497	13	0.0009
smallinvDAXr5b150-165	17.016	51	0.0010	10.465	19	0.0010	6.072	11	0.0009
smallinvDAXr5b200-220	15.207	47	0.0010	10.104	18	0.0009	7.396	11	0.0009
squfl025-030	900*	266*	0.4438*	900*	136*	0.6127*	900*	150*	0.6333*
squfl025-025	900*	267*	0.4102*	900*	118*	0.6177*	900*	133*	0.7265*
squfl010-025	132.307	145	0.0000	116.474	94	0.0007	123.704	94	0.0000
fac2	2.544	7	0.0000	3.546	7	0.0000	3.108	6	0.0000
fac3	2.347	7	0.0000	3.331	7	0.0000	2.965	6	0.0000
fac1	1.469	3	0.0000	1.171	3	0.0000	1.199	3	0.0000
ex1223	1.483	5	0.0000	2.181	5	0.0000	2.147	5	0.0000
st_e14	1.456	5	0.0000	2.174	5	0.0000	2.154	5	0.0000
batchdes	0.605	2	0.0000	0.612	2	0.0000	0.599	2	0.0000
cvxnonsep_pcon20r	1.816	7	0.0002	4.207	9	0.0003	1.942	5	0.0008
cvxnonsep_pcon30r	1.899	7	0.0002	4.474	9	0.0006	3.191	7	0.0007
cvxnonsep_pcon40r	1.607	6	0.0008	4.99	10	0.0006	4.484	9	0.0010
cvxnonsep_psig40r	1.273	5	0.0007	4.853	9	0.0003	2.215	5	0.0003
cvxnonsep_normcon40r	3.112	7	0.0000	5.694	9	0.0007	2.974	5	0.0007
cvxnonsep_nsig40r	2.341	8	0.0005	7.702	13	0.0007	2.589	6	0.0007
cvxnonsep_psig30r	1.855	7	0.0003	5.055	10	0.0002	2.528	6	0.0008
cvxnonsep_normcon30r	3.203	9	0.0000	6.422	12	0.0005	3.577	7	0.0009
cvxnonsep_nsig30r	1.645	5	0.0007	4.95	10	0.0006	2.39	6	0.0007
cvxnonsep_psig20r	1.831	6	0.0006	7.545	15	0.0008	2.486	6	0.0008
cvxnonsep_normcon20r	1.913	6	0.0000	5.456	11	0.0000	4.562	10	0.0009
cvxnonsep_nsig20r	1.449	5	0.0009	4.906	10	0.0006	2.211	5	0.0006
st_miqp4	0.386	2	0.0000	0.369	2	0.0000	0.375	2	0.0000
ibs2	900*	431*	0.0054*	900*	41*	0.1093*	103.888	16	0.0010