Improved quadratic cuts for convex mixed-integer nonlinear programs

Lijie Su[a,b], Lixin Tang[a*], David E. Bernal[c] , Ignacio E. Grossmann[c]

[a] Institute of Industrial and Systems Engineering, Northeastern University, Shenyang 110819, P. R. China

[b] State Key Laboratory of Synthetical Automation for Process Industries, Northeastern University, Shenyang, 110819, P. R. China

[c] Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh 15213, PA, USA

Abstract

This paper presents scaled quadratic cuts based on scaling the second-order Taylor expansion terms for the decomposition methods Outer Approximation (OA) and Partial Surrogate Cuts (PSC) used for solving convex Mixed Integer Nonlinear Programing (MINLP). The scaled quadratic cut is proved to be a stricter and tighter underestimation for the convex nonlinear functions than the classical supporting hyperplanes which results in the improvement of efficiency of the OA and PSC based MINLP solution methods. We integrate the strategies of the scaled quadratic cuts with multi-generation cuts with OA and PSC, and develop six types of MINLP solution methods with scaled quadratic cuts. This improvement transforms the master problem of the decomposition methods in a Mixed Integer Quadratically Constrained Programming (MIQCP) problem. Numerical results of benchmark MINLP problems demonstrate the effectiveness of the proposed MINLP solution methods with scaled quadratic cuts.

* Corresponding author. *E-mail address*: Lixintang@mail.neu.edu.cn (Lixin Tang)

Key words

MINLP; Outer Approximation (OA); Quadratic cut; MIQCP

1. Introduction

Mathematical programming models can be written in general form including linear and nonlinear functions, and continuous and discrete variables as MINLPs. The applications of MINLP models in Process Systems Engineering (PSE) are very extensive. The MINLP models are usually used to formulate problems such as synthesis, cyclic production planning, batch scheduling, operations optimization, optimal control, among others (Méndez et al., 2006; Tang et al., 2002, n.d.; Trespalacios and Grossmann, 2014). The nature of the functions involved in optimization in PSE motivated the development of MINLP solution methods.

Mathematical programming models in PSE include Linear Programming (LP), Integer Programming (IP), and Mixed Integer Linear Programming (MILP) on one side, and Nonlinear Programming (NLP) on the other; all of which can be described with MINLP.

The most common methods for solving convex MINLP problems include decomposition methods, such as Outer Approximation (OA), Generalized Benders Decomposition (GBD), Extended Cutting Planes (ECP), and Branch and Bound (B&B) methods. Methods for nonconvex MINLP problems include mostly spatial B&B methods (Tawarmalani and Sahinidis, 2004). Reviews at MINLP methods can be found in Grossmann (2002), Bonami et al. (2012) and Belotti et al. (2013).

A general form of an MINLP is as follows.

$$
\begin{aligned}
Z = \min_{x,y} \quad & f(x,y) \\
s.t. \quad & g(x,y) \le 0 \\
& Ax + Ey \le b \\
& x \in X \subset \mathbb{R}^n \\
& y \in Y \subset \mathbb{Z}^p
\end{aligned} \qquad (P)
$$

where $f : \mathbb{R}^n \times \mathbb{R}^p \to \mathbb{R}$ is the objective function which may be nonlinear, and $g : \mathbb{R}^n \times \mathbb{R}^p \to \mathbb{R}^m$ are the nonlinear inequality constraints. The matrices A and E are the coefficient matrices of the continuous and discrete variables in the linear constraints, respectively, and the vector b is the right hand side of the linear inequalities. The sets X and Y are compact and bounded subsets of the $n-$dimensional Euclidean space and the $p-$dimensional integer lattice, respectively. The optimal solution to the problem, if it exists, is denoted by (x^*, y^*).

The discrete nature of the y variables makes the MINLP problems nonconvex, but a usual classification of MINLP models is based on the convexity of the objective function $f(x,y)$ and constraints $g(x,y)$. If the objective function and the constraints are all convex, the MINLP can be classified as convex, and nonconvex otherwise.

This paper addresses the solution of convex MINLP based on OA and PSC methods using scaled quadratic cuts that underestimate the convex nonlinear functions, and it is organized as follows. In Section 2, we present background in MINLP solution methods, with a special focus on the OA and the PSC methods and the multi-generation cut strategy. Section 3 presents a motivation for this work, where quadratic approximation to nonlinear functions can approximate more accurately the MINLP. Section 4 presents the quadratic approximation cuts, and introduces the scaled quadratic cuts, which are proven to be valid underestimators of the convex nonlinear functions. In Section 5 we present the improved MINLP methods based on the scaled quadratic cuts, including the multi-generation strategies and PSC methods; finally we present six new convex MINLP algorithms. Section 6 presents

the numerical experiments of applying the proposed algorithms to 44 benchmark MINLP problems, and comparing their performances among then and with existing MINLP solvers. Finally, we give some conclusions for the improved OA and PSC methods with the proposed quadratic cuts.

2. Background

In the following section, we summarize the different solution methods for MINLP, the Outer-Approximation (OA) method, the Partial Surrogate Cut (PCS) method, and the multi-generation cuts strategy. The following assumptions are made:

(A1) The function $f:\mathbb{R}^n\times\mathbb{R}^p\to\mathbb{R}$ and vector functions $g:\mathbb{R}^n\times\mathbb{R}^p\to\mathbb{R}^m$ are twice continuously differentiable convex functions, in which there is at least one nonlinear function.

(A2) x and y represent the continuous and discrete variables, respectively. The set X is a nonempty compact convex set, and the set Y finite.

(A3) All NLP subproblems with fixed discrete variables of MINLP satisfy a constraint qualification (Biegler, 2010), which is the precondition for the correct solution of the MINLP problem.

Here, the discrete variables y can be regarded as 0,1 binary variables. General bounded discrete variables can always be expressed in terms of binary variables.

2.1. MINLP solution methods

The general solution methods for convex MINLP problems are classified into two categories according to the structure of the algorithmic procedure. The first category of MINLP methods are the decomposition methods, which includes OA (Duran and Grossmann, 1986), GBD (Geoffrion, 1972), and PSC (Su et al., 2015). The second category of MINLP methods are based on B&B (Gupta and

Ravindran, 1985) including LP/NLP based B&B (Bonami et al., 2012; Leyffer, 2001; Quesada and Grossmann, 1992). The convex MINLP solution methods are used within global optimization algorithms for finding the global optimal solution to nonconvex MINLP problems (Adjiman et al., 1998; Kesavan et al., 2004; Li et al., 2011; Tawarmalani and Sahinidis, 2004).

The idea of decomposition methods for convex MINLP is to decompose an MINLP into an NLP with fixed discrete variables and an MILP based on constructing cumulative relaxation cuts of nonlinear functions, which are the underestimators of the convex nonlinear functions.

The NLP subproblem of an MINLP in minimization problems yields an upper bound assuming a feasible solution exists. The problem for fixed integer variables y^k is as follows.

$$\begin{aligned} Z_{UB}^k = \min_x \quad & f\left(x, y^k\right) \\ \text{s.t.} \quad & g\left(x, y^k\right) \le 0 \\ & Ax + Ey^k \le b \\ & x \in X \subset \mathbb{R}^n \end{aligned} \qquad (NLP(y^k))$$

Let (x^k) be the solution to the $NLP(y^k)$ problem, if feasible. Note that the solution (x^k, y^k) is a feasible solution to the problem P. The MILP problem based on relaxation cuts of the nonlinear functions is called the master problem M^k shown later in this paper, and yields a lower bound for minimization problems. By successively solving the NLP and the MILP in an iterative cycle, upper and lower bounds of the objective function are obtained, and the procedure is stopped when the bounds lie within a given tolerance. The differences of the decomposition methods lie on the MILP master problem, especially in the process of generating the relaxation cuts, which also determine the problem size of MILP and the predicted lower bounds for the MINLP.

Numerical experiments on MINLP benchmark problems demonstrate that decomposition methods usually require shorter CPU times than B&B methods (Adjiman et al., 1998; Su et al., 2015).

Because of the accumulated cuts in the MILP master problem of decomposition methods, the problem size can become significantly large, which may lead to longer computational time. Moreover, the convergence of decomposition methods may become slow when the approximating cuts are not very tight, leading to weaker lower bounds (Su et al., 2015).

2.2. Outer-approximation method

The OA method is a decomposition method proposed by Duran and Grossmann (1986) for solving MINLP problems. In this method, the master problem is constructed with the accumulated linear cuts on the optimal solutions of NLP subproblems through the first-order Taylor expansion of nonlinear functions. The master problem is given by the following MILP:

$$
\begin{aligned}
& Z_{LB,OA}^{k} = \min_{\eta,x,y} \quad \eta \\
& s.t. \quad \eta \geq f(x^k, y^k) + \nabla f(x^k, y^k)^T \begin{pmatrix} x - x^k \\ y - y^k \end{pmatrix} \quad \forall k \in K \\
& \qquad g(x^k, y^k) + \nabla g(x^k, y^k)^T \begin{pmatrix} x - x^k \\ y - y^k \end{pmatrix} \leq 0 \quad \forall k \in K \\
& \qquad Ax + Ey \leq b \qquad\qquad (M_{OA}^{k}) \\
& \qquad \eta \in \mathbb{R}^1 \\
& \qquad x \in X \subset \mathbb{R}^n \\
& \qquad y \in Y \subset \mathbb{Z}^p
\end{aligned}
$$

The OA method decomposes a convex MINLP problem into an NLP subproblem $NLP(y^k)$ and an MILP master problem M_{OA}^{k}. The feasible solutions of $NLP(y^k)$ are feasible solutions of P, and therefore correspond to upper bounds of the original problem,. The solutions of the MILP are non-decreasing lower bounds of the MINLP. From a geometric point of view, an OA cut is the tangent passing through one active nonlinear function in the MINLP problem.

For the first iteration of the OA method, Viswanathan and Grossmann (1990) proposed that the first linearizations of the problem be generated based on the solution of the relaxed MINLP, in which the

integer variables are treated as continuous,

$$\begin{aligned} Z = \min_{x,y} \quad & f(x,y) \\ s.t. \quad & g(x,y) \le 0 \\ & Ax + Ey \le b \\ & x \in X \subset \mathbb{R}^n \\ & y \in Y \subset \mathbb{R}^p \end{aligned} \qquad (rMINLP)$$

The relaxed MINLP is an NLP problem that allows the user not to provide an initial integer vector. In case that the optimal solution of this problem satisfies the integrality constraints, no iterations are required as then the solution is found. In the case that this problem is infeasible and unbounded, the same can be assured for the original MINLP problem (Viswanathan and Grossmann, 1990).

In order to avoid visiting the same integer combination of variables, an integer cut can be added to the master problem. For the case of binary variables, the following constraint is a valid cut (Duran and Grossmann, 1986).

$$\sum_{j \in B^k} y_j - \sum_{j \in NB^k} y_j \le |B^k| - 1 \qquad (1)$$

where $B^k = \{j : y_j^k = 1\}$, $NB^k = \{j : y_j^k = 0\}$ and $|B| + |NB| = p$.

When fixing the integer variables y^k there is a possibility that the $NLP(y^k)$ problem becomes infeasible. Even in this case, the values of the continuous variables x can be used to generate valid linearizations if the functions are convex (Quesada and Grossmann, 1992). This point can be avoided in following iterations using the integer cut in Eq. (1). Furthermore, Fletcher and Leyffer (1994) proposed to solve the following feasibility NLP subproblem after finding an infeasible $NLP(y^k)$ to minimize the violation of the constraints,

$$
\begin{aligned}
\min_{x,s} \quad & \sum_{i=1}^{m} s_i \\
s.t. \quad & g(x,y) \le s \\
& Ax + Ey \le b \\
& s \in \mathbb{R}_{+}^{m} \\
& x \in X \subset \mathbb{R}^{n}
\end{aligned}
\qquad (F(y^k))
$$

2.3. Partial surrogate cuts method

Another decomposition method is the PSC method, proposed by Quesada and Grossmann (1992). In this method, the main idea is to partition the continuous variables into linear and nonlinear according to their presence in linear and nonlinear terms, respectively. This allows us to rewrite the problem P as follows,

$$
\begin{aligned}
Z = \min_{x,y} \quad & f(v,w,y) \\
s.t. \quad & g(v,w,y) \le 0 \\
& A_1 w + A_2 v + Ey \le b \\
& \begin{pmatrix} w \\ v \end{pmatrix} = x \in X \subset \mathbb{R}^{n} \\
& y \in Y \subset \mathbb{Z}^{p}
\end{aligned}
\qquad (P')
$$

where w are the linear and v are the nonlinear continuous variables, respectively. The coefficient matrix of the continuous variables in the linear constraints is also partitioned into A_1 and A_2 according to the linear and nonlinear continuous variables, respectively. This partition allows to build a Lagrangean cut by projecting out the nonlinear terms as follows.

$$
\eta \ge f(v^k,w,y) + \begin{pmatrix} \lambda^k \\ -\mu^k \end{pmatrix}^T \begin{bmatrix} g(v^k,w,y) & 0 \\ 0 & A_2 \end{bmatrix} \begin{pmatrix} 1 \\ v - v^k \end{pmatrix} \qquad (2)
$$

where the λ^k are the Lagrange multipliers of the nonlinear constraints including the nonlinear continuous variables v, μ^k are the Lagrange multipliers of the linear equations including nonlinear continuous variables v, both obtained from the solution of the $rMINLP$ problem at the first iteration $k=1$ and the $NLP(y^k)$ problem at iteration $k \ge 2$.

Contrary to OA, the linearizations based on the projection onto the nonlinear terms are not valid when the solution of the $NLP(y^k)$ problem is infeasible, in which case the following infeasibility cut is applied from the dual information of the nonlinear subproblem.

$$\begin{pmatrix} \lambda^k \\ -\mu^k \end{pmatrix}^T \begin{bmatrix} g\left(v^k, w, y\right) & 0 \\ 0 & A_2 \end{bmatrix} \begin{pmatrix} 1 \\ v - v^k \end{pmatrix} \leq 0 \tag{3}$$

With these cuts the master problem of the PSC method is as follows:

$$\begin{aligned} Z^k_{LB,PSC} = \min_{x,y} \quad & \eta \\ s.t. \quad & \eta \geq f\left(v^k, w, y\right) + \begin{pmatrix} \lambda^k \\ -\mu^k \end{pmatrix}^T \begin{bmatrix} g\left(v^k, w, y\right) & 0 \\ 0 & A_2 \end{bmatrix} \begin{pmatrix} 1 \\ v - v^k \end{pmatrix} \quad \forall k \in K_F \\ & \begin{pmatrix} \lambda^k \\ -\mu^k \end{pmatrix}^T \begin{bmatrix} g\left(v^k, w, y\right) & 0 \\ 0 & A_2 \end{bmatrix} \begin{pmatrix} 1 \\ v - v^k \end{pmatrix} \leq 0 \quad \forall k \in K_I \qquad (M^k_{PSC}) \\ & \eta \in \mathbb{R}^1 \\ & x \in X \subset \mathbb{R}^n \\ & y \in Y \subset \mathbb{Z}^p \end{aligned}$$

where K_F and K_I are the set of iterations where the $NLP(y^k)$ problem was feasible and infeasible, respectively.

The PSC method can be seen as a combination of the OA and the GBD methods, where the cuts are derived from the gradient based linearizations of OA, and the KKT conditions as in GBD. The relationship between the lower bounds in minimization MINLP problems for each method is as follows (Quesada and Grossmann, 1992):

$$Z^k_{LB,GBD} \leq Z^k_{LB,PSC} \leq Z^k_{LB,OA} \tag{4}$$

The size of the master problem from outer approximation is larger than the one is the PSC method, although it provides stronger lower bounds. This tradeoff between the methods allows to choose among them depending on the problem, e.g. if the M^k_{OA} problem becomes too large to handle, the PSC method can become advantageous at the cost of having to perform more iterations to find the

optimal solution of the MINLP problem (Su et al., 2015).

2.4. Multi-generation cuts

In decomposition methods for solving MINLP problems, the master problem M^k accumulates cuts that improve its approximation to the original problem P. The original OA, GBD and PSC methods generate a single cut per iteration, but Su et al. (2015) proposed to add several cuts at every iteration. The addition of the several cuts can be made using the solution pool of the MILP solver, such as CPLEX (IBM Corp. and IBM, 2009), while solving the master problem M^k and solving a $NLP(y^k)$ subproblem for each feasible solution in the solution pool.

Given S feasible solutions of the master problem M^k, we solve for each one a subproblem $NLP(y^{k,s})$ with $s = 1,2,...,|S|$. The solution to each problem, $(x^{k,s}, y^{k,s})$, will be a valid point for generating linearizations for a convex MINLP.

The OA cuts generated at the k^{th} iteration are as follows.

$$\begin{aligned} \eta &\geq f(x^{k,s}, y^{k,s}) + \nabla f(x^{k,s}, y^{k,s})^T \begin{pmatrix} x - x^{k,s} \\ y - y^{k,s} \end{pmatrix} \quad s \in S \\ g(x^{k,s}, y^{k,s}) &+ \nabla g(x^{k,s}, y^{k,s})^T \begin{pmatrix} x - x^{k,s} \\ y - y^{k,s} \end{pmatrix} \leq 0 \quad s \in S \end{aligned} \tag{5}$$

In a similar fashion, multiple cuts can be generated for the PSC method as follows.

$$\begin{aligned} \eta &\geq f\left(v^{k,s}, w, y\right) + \begin{pmatrix} \lambda^{k,s} \\ -\mu^{k,s} \end{pmatrix}^T \begin{bmatrix} g\left(v^{k,s}, w, y\right) & 0 \\ 0 & A_2 \end{bmatrix} \begin{pmatrix} 1 \\ v - v^{k,s} \end{pmatrix} \quad \forall k \in K_F, s \in S \\ &\begin{pmatrix} \lambda^{k} \\ -\mu^{k} \end{pmatrix}^T \begin{bmatrix} g\left(v^{k}, w, y\right) & 0 \\ 0 & A_2 \end{bmatrix} \begin{pmatrix} 1 \\ v - v^{k} \end{pmatrix} \leq 0 \quad \forall k \in K_I, s \in S \end{aligned} \tag{6}$$

The multi-generation cuts (MC) procedure strengthens the approximation of the master problem, and therefore the lower bound in minimization MINLP problems can be improved. This approach can reduce the number of iterations of decomposition algorithms such as OA and PSC. Even though the

iterations can be more time-consuming, the total computational time can decrease using this procedure (Su et al., 2015).

3. Motivation

Quadratic approximations have been proposed for MINLP decomposition methods and B&B methods. Grossmann (2002) and Fletcher and Leyffer (1994) discussed the quadratic master problem of decomposition methods. Based on the Hessian of the Lagrange function with respect to continuous and discrete variables, the master problem is constructed with a quadratic objective function and linear constraints which results in a Mixed-Integer Quadratic Programming (MIQP) master problem. One drawback of this approach is that at the time there was no software available to efficiently solve MIQP problems (Fletcher and Leyffer, 1994).

Recently, MILP solvers like CPLEX and Gurobi have expanded their capabilities to solve MIQP and Mixed-Integer Quadratically Constrained Programming (MIQCP) problems (Bliek et al., 2014). The crucial factor of MIQCP solvability is the Hessian matrix. Convex MIQCP problem, which have a positive semi-definite Hessian matrix, can be solved to optimality as one special convex MINLP with the MILP solver CPLEX (IBM Corp. and IBM, 2009) and with the Barrier method for QCP subproblem (Nesterov and Todd, 1998).

Buchheim and Trieu (2013) presented a quadratic outer approximation scheme for convex integer nonlinear programming problems, which are problems with an unconstrained convex nonlinear objective function with only integer variables. Leyffer (2001) integrated the Sequential Quadratic Programming (SQP) algorithm in the branch and bound method.

The relaxation of the nonlinear constraints using linearizations can also be further improved by

considering quadratic approximation cuts. In the MINLP decomposition methods with quadratic approximation cuts, the master problem is a MIQCP. If the objective function and constraints in MIQCP are convex, the MIQCP optimization problem is convex, which can be solved using solution methods for convex MINLP. If the MIQCP can be transformed into Mixed Integer Second Order Cone Programming (MISOCP), it can be efficiently solved using Second Order Cone Programming (SOCP) solution techniques (Alizadeh and Goldfarb, 2003). Alternatively, Berthold et al. (2012) presented a global solution framework based on constraint integer programming, that was used as a basis for the SCIP solver, that is aimed at the solution of MIQCP problems with general Hessian matrix (Achterberg, 2009).

If the quadratic cuts based on Taylor 2^{nd} order term are introduced into the decomposition MINLP methods, the objective function can be better approximated than with the general linear master problem. However, the quadratic Taylor expansion does not provide a rigorous lower bound for general nonlinear functions. Hence, the OA and PSC with an MIQCP master problem is not guaranteed to converge to the optimal solution of the MINLP. Nevertheless, it is interesting to study whether the quadratic approach for the OA and PSC method can be made rigorous, and help to accelerate the solution process.

We replace the tangent cuts of the master problem with scaled quadratic cuts that underestimate the convex nonlinear functions. These scaled quadratic cuts allow us to use second order information and still ensure that we underestimate the nonlinear functions with an approximation at most as nonlinear as the 2^{nd} order Taylor expansion. In combination with the strategy of multi-generation cut, we present the strategy of multi-generation quadratic cuts and hybrid cuts for OA and PSC for solving convex MINLP problems. Numerical experiments demonstrate the performance of the improved OA and PSC

method.

4. Quadratic approximation cuts

The capabilities of solving efficiently MIQCP problems has allowed us to use second order derivative information to build a closer approximation to the nonlinear constraints in MINLP problems. In this section, we show how using Taylor series truncated at the second order we can build tighter estimators for the nonlinear functions. We then show how using a scale factor α we can ensure that our quadratic cuts (QCUT), are valid underestimators for the nonlinear functions. Finally, based on these observations, we obtain QCUT which are valid underestimators for the convex nonlinear functions involved.

4.1. Quadratic approximations of nonlinear functions

According to Taylor's theorem, any function $f:\mathbb{R}^n \to \mathbb{R}$ of the class C^{n+1} on the open convex set B containing the point x_0 can be written as an expansion around x_0 using a polynomial whose coefficients depend in the derivatives of the function in that point as follows (Folland, 2002).

$$f(x) = \sum_{|\gamma|=0}^{n} \frac{1}{\gamma!} \frac{\partial^{\gamma} f(x_0)}{\partial x^{\gamma}} (x - x_0)^{\gamma} + R_{x_0,n}(x - x_0) \tag{7}$$

where γ is an *n*-tuple of nonnegative integers used as a compressed notation to the multi-dimensional Taylor's theorem. In this case we define

$$\begin{aligned} &\gamma = (\gamma_1, \gamma_2, \ldots, \gamma_n) \\ &\gamma! = \gamma_1! \gamma_2! \ldots \gamma_n! \\ &|\gamma| = (\gamma_1 + \gamma_2 + \ldots + \gamma_n) \\ &x^{\gamma} = x_1^{\gamma_1} x_2^{\gamma_2} \ldots x_n^{\gamma_n} \\ &\partial^{\gamma} f = \partial_1^{\gamma_1} \partial_2^{\gamma_2} \ldots \partial_n^{\gamma_n} f = \frac{\partial^{|\gamma|} f}{\partial x_1^{\gamma_1} \partial x_2^{\gamma_2} \ldots \partial x_n^{\gamma_n}} \end{aligned} \tag{8}$$

The polynomial generated using the derivative information is called the Taylor expansion, and the

number $|\gamma| = (\gamma_1 + \gamma_2 + \ldots + \gamma_n)$ is defined as the order or degree of γ. The term $R_{x_0,n}(x - x_0)$ is known as the remainder of the Taylor expansion.

Note that the 1st order Taylor expansion can be defined using the gradient of the function at the expansion point $\nabla f(x_0)$ as follows.

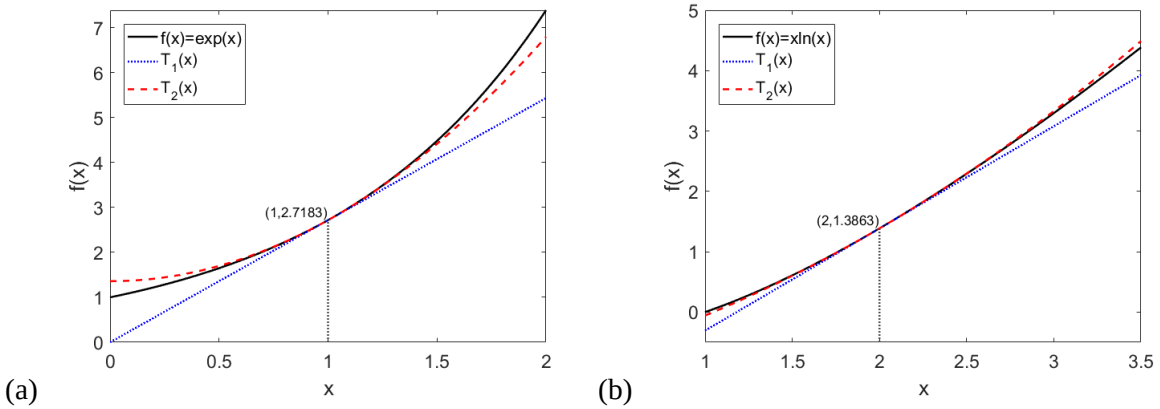
$$T_1(x) = f(x_0) + \nabla f(x_0)^T (x - x_0) \tag{9}$$

When we expand a nonlinear function with 2nd-order terms of Taylor series, we obtain the quadratic approximation of the function.

$$T_2(x) = f(x_0) + \nabla f(x_0)^T (x - x_0) + \frac{1}{2}(x - x_0)^T \nabla^2 f(x_0)(x - x_0) \tag{10}$$

where the $\nabla^2 f(x_0)$ denotes the Hessian matrix of the function f evaluated in the point x_0.

The advantage of the 2nd-order Taylor expansion for a convex function is that it provides a better approximation to the nonlinear function than the 1st-order Taylor expansion in the neighborhood of the expanded point x_0, although it may not always be an underestimation of the original function as seen in Figure 1.



(a) (b)

Figure 1. Comparisons of the 1st and 2nd-order Taylor expansions of (a) e^x at $x_0 = 1$ and defined for $0 \le x \le 2$ and (b) $x \ln x$ at $x_0 = 2$ and defined for $1 \le x \le 3.5$

The comparisons of the 1st- and 2nd-order Taylor expansions for the functions e^x and $xln(x)$ at a

fixed point x_0 are illustrated in Figure 1. The deviation between the nonlinear function and the 2nd-order Taylor expansion is less than the deviation between nonlinear function and the 1st-order Taylor expansion. We can than state the following proposition.

Proposition 1. If a convex nonlinear function is expanded in one fixed point, the 2nd-order Taylor expansion of the function must be greater than or equal to the 1st-order Taylor expansion for any point.

Proof. For a convex function $f(x)$, the 2nd-order Taylor expansion $T_2(x)$ minus the 1-order Taylor expansion $T_1(x)$ is given by

$$\Delta T(x) = \frac{1}{2}(x - x_0)^T \nabla^2 f(x_0)(x - x_0) \tag{11}$$

If the nonlinear function is convex, $\nabla^2 f(x_0)$ is positive semi-definite. Therefore, for any point *x*,

$$\Delta T(x) = \frac{1}{2}(x - x_0)^T \nabla^2 f(x_0)(x - x_0) \geq 0 \tag{12}$$

Therefore, the 2nd-order Taylor expansion of the convex function must be greater than or equal to the 1st-order Taylor expansion at any point. □

From the geometric point of view, the curve of 2nd-order Taylor expansion must lie over the tangent line of the 1st-order Taylor expansion. However, the drawback is that the 2nd-order Taylor expansion of one function is not always a valid underestimation of the nonlinear function, as is the case of the 1st-order expansion. The 2nd-order Taylor expansions shown in Figure 1, are not the underestimations of the convex functions e^x and $x \ln x$. Therefore, in the next section we propose a scaled quadratic cut, which provides a valid underestimation of convex functions.

4.2. Scaled quadratic approximations

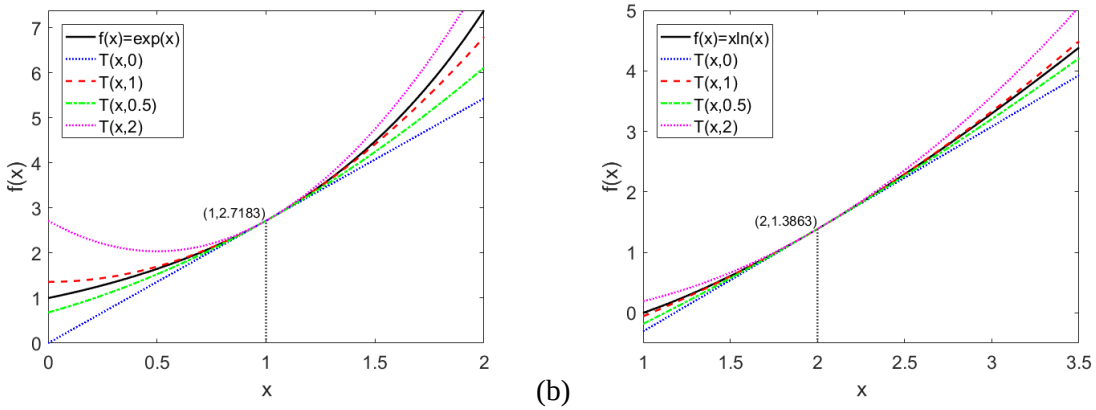
In order to globally underestimate a convex nonlinear function by a quadratic cut, we expand the 2nd-order Taylor expansion to a quadratic approximation with a scaled coefficient α in [0, 1] as in

Eq. (13).

$$T(x,\alpha) = f(x_0) + \nabla f(x_0)^T (x - x_0) + \frac{1}{2}\alpha (x - x_0)^T \nabla^2 f(x_0)(x - x_0) \tag{13}$$

Note that if α is equal to 1, $T(x,\alpha)$ corresponds to the 2nd-order Taylor expansion, whereas if α is equal to 0 it corresponds to the 1st-order Taylor expansion. When we set the range of α as $0 \le \alpha \le 1$, $T(x,\alpha)$ corresponds to an approximation that lies between the supporting hyperplane and the quadratic expansion, which still is a quadratic approximation with a larger radius of curvature at x_0. On the other hand, for the range of $\alpha \ge 1$, $T(x,\alpha)$ is an overestimation of the 2nd order Taylor expansion.

For the convex functions considered in Figure 1, we add scaled quadratic cuts as shown in Figure 2. We can notice how the scaled 2nd-order Taylor expansion lies between the 1st and 2nd-order Taylor expansions when α is between 0 and 1, while with $\alpha > 1$ it is above the 2nd order approximation.



(a) (b)

Figure 2. Comparisons of the 1st and 2nd-order scaled Taylor expansions with different values of α of (a) e^x at $x_0 = 1$ and (b) $x \ln x$ at $x_0 = 2$

Therefore, if the coefficient α satisfies certain conditions, it can yield quadratic approximations that are underestimations of the convex nonlinear functions. The following propositions provide theoretical guarantees assuming that the variables are bounded.

Proposition 2. Let the general convex twice continuously differentiable function $f(x)$ of the n-dimensional vector x be bounded by $x \in [a,b]$. If we calculate the scalar α in Eq. (13) as follows,

$$\frac{\alpha}{2} = \min_{x \in F = \{x | a \le x \le b\} \setminus \{x_0\}} \left\{ \frac{f(x) - f(x_0) - \nabla f(x_0)^T (x - x_0)}{(x - x_0)^T \nabla^2 f(x_0)(x - x_0)} \right\} \tag{14}$$

Then the scaled 2nd order Taylor approximation $T(x,\alpha)$ is a valid underestimator of the function $f(x)$.

Proof. Let $f : \mathbb{R}^n \to \mathbb{R}$ be a convex, twice continuously differentiable function defined over the set $a \le x \le b$, where a, b are finite bounds. Let $T(x,\alpha)$ be the scaled 2nd Taylor approximation, and α be scaled coefficient for quadratic approximation as in Eq. (13).

The necessary condition to prove in this proposition is the following.

$$f(x) \ge T(x,\alpha) \quad \forall x \in F = \{x | a \le x \le b\} \tag{15}$$

If we define the function $g(x)$ as the difference between the original function and the scaled approximation, we have the following condition to prove.

$$g(x) = f(x) - T(x,\alpha) \ge 0 \quad \forall x \in F = \{x | a \le x \le b\} \tag{16}$$

Replacing Eq. (13) we obtain the following condition,

$$f(x) - \nabla f(x_0)^T (x - x_0) - \frac{1}{2}\alpha (x - x_0)^T \nabla^2 f(x_0)(x - x_0) \ge 0 \tag{17}$$

Given the fact that $f(x)$ is convex and from Proposition 1, we can rearrange Eq. (17) as follows.

$$\frac{f(x) - \nabla f(x_0)^T (x - x_0)}{(x - x_0)^T \nabla^2 f(x_0)(x - x_0)} \ge \frac{\alpha}{2} \quad \forall x \ne x_0 \tag{18}$$

Since the term in the denominator is greater or equal than zero it does not change the inequality sign and we can exclude the expansion point from the domain of the inequality. Using now Eq. (14), we obtain the following.

$$\frac{f(x)-\nabla f(x_0)^T(x-x_0)}{(x-x_0)^T\nabla^2 f(x_0)(x-x_0)} \geq \min_{x\in F=\{x|a\leq x\leq b\}\setminus\{x_0\}} \left\{ \frac{f(x)-f(x_0)-\nabla f(x_0)^T(x-x_0)}{(x-x_0)^T\nabla^2 f(x_0)(x-x_0)} \right\} \tag{19}$$

which by definition of the minimizer operator, must hold. This shows that the scaled 2nd order Taylor approximation $T(x,\alpha)$ calculating the α using Eq. (14) is a valid underestimator of the convex function $f(x)$. □

Given the fact that the function $f(x)$ is convex, the term $(x-x_0)^T\nabla^2 f(x_0)(x-x_0)$ is greater or equal than zero, and therefore the function $g(x)$ is monotonic decreasing with respect to α. This means that minimizing the value of the scalar maximizes the difference between $f(x)$ and $T(x,\alpha)$.

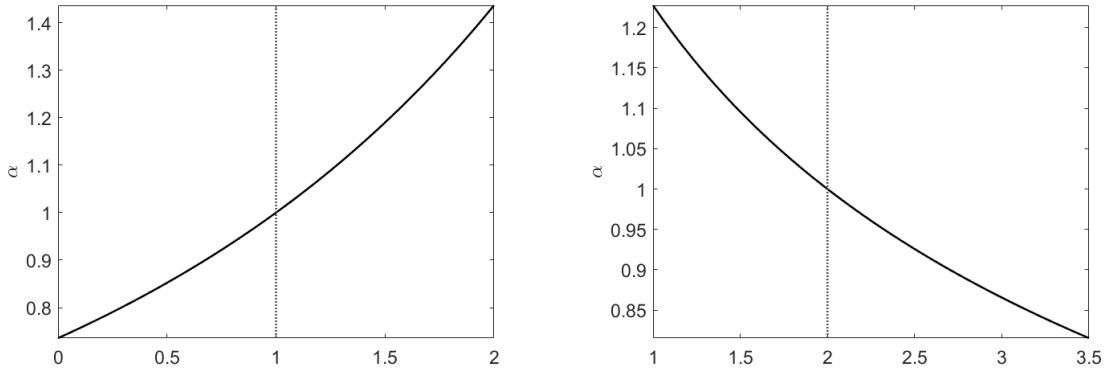
The fact of having a convex $f(x)$ does not mean that the function defining α is convex itself. Therefore, finding its minimum is not trivial, although one observation can avoid solving this nonconvex optimization problem.

Let *V* be the set of vertices $x^v \in V$ over those bounds. The function $f(x)$ is convex, and therefore $T(x,\alpha)$ is convex too, hence if we maximize those functions in a closed region F, its solution must be at a vertex $x^v \in V$ in the feasible set *F.* This can be seen in Theorem I.1 of Chapter 2 of Horst and Tuy, 2013.

Therefore, we can assume that the maximum of the difference between these functions, $g(x)$ lies at one of the vertices. In this case the function defining α is coordinate-wise monotonic, i.e. nondecreasing or nonincreasing in each variable independently and implies that the minimum α can be found using the following equation.

$$\frac{\alpha}{2} = \min_{x^v\in V} \left\{ \frac{f(x^v)-f(x_0)-\nabla f(x_0)^T(x^v-x_0)}{(x^v-x_0)^T\nabla^2 f(x_0)(x^v-x_0)} \right\} \quad \forall x_0 \neq x^v \tag{20}$$

This expression can be evaluated for all the vertices $x^v \in V$, and the minimum can be found by exhaustive enumeration instead of solving a nonconvex optimization problem. The function describing α is the ratio between the difference of the original function and its 1st-order Taylor approximation and the quadratic term in the 2nd-order Taylor approximation. Eq. (20) considers that if this function is component-wise monotonic, the geometrical interpretation is that the minimum of this function will be in one of the vertices as seen in Figure 3.



(a) (b)

Figure 3. Function calculating the scalar parameter α for the functions (a) e^x at $x_0 = 1$ and defined for $0 \le x \le 2$ and (b) $x \ln x$ at $x_0 = 2$ and defined for $1 \le x \le 3.5$

Although this condition of coordinate-wise monotonicity in α is stronger than the convexity of the function $f(x)$, the problems used in this work satisfy that the minimum α is indeed in a vertex. More details regarding this are included in the Appendix.

In summary, based on Proposition 2 we generate the quadratic underestimations of convex nonlinear functions for MINLP problems in order to accelerate the convergence of the OA and PSC algorithms.

4.3. Scaled quadratic cuts for Outer-approximation

We generate the scaled quadratic cuts in the master problem of OA as follows.

$$\begin{aligned}
&\eta \geq f(x^k, y^k) + \nabla f(x^k, y^k)^T \begin{pmatrix} x - x^k \\ y - y^k \end{pmatrix} + \frac{\alpha^k}{2} \begin{pmatrix} x - x^k \\ y - y^k \end{pmatrix}^{\mathrm{T}} \nabla^2 f(x^k, y^k) \begin{pmatrix} x - x^k \\ y - y^k \end{pmatrix} \quad \forall k \\
&g(x^k, y^k) + \nabla g(x^k, y^k)^T \begin{pmatrix} x - x^k \\ y - y^k \end{pmatrix} + \frac{\beta^k}{2} \begin{pmatrix} x - x^k \\ y - y^k \end{pmatrix}^{\mathrm{T}} \nabla^2 g(x^k, y^k) \begin{pmatrix} x - x^k \\ y - y^k \end{pmatrix} \leq 0 \quad \forall k
\end{aligned} \tag{21}$$

where α^k is a scaled coefficient of objective function, β^k is a scaled vector of the constraint functions vector for the k^{th} iteration of OA. The scalar α^k and scaled vector β^k are calculated based on Eq. (20).

The master problem then becomes the following MIQCP problem.

$$\begin{aligned}
&Z^k_{LB,OA-QCUT} = \min_{\eta,x,y} \quad \eta \\
&s.t \quad \eta \geq f(x^k, y^k) + \nabla f(x^k, y^k)^T \begin{pmatrix} x - x^k \\ y - y^k \end{pmatrix} + \frac{\alpha^k}{2} \begin{pmatrix} x - x^k \\ y - y^k \end{pmatrix}^{\mathrm{T}} \nabla^2 f(x^k, y^k) \begin{pmatrix} x - x^k \\ y - y^k \end{pmatrix} \quad \forall k \in K \\
&\quad g(x^k, y^k) + \nabla g(x^k, y^k)^T \begin{pmatrix} x - x^k \\ y - y^k \end{pmatrix} + \frac{\beta^k}{2} \begin{pmatrix} x - x^k \\ y - y^k \end{pmatrix}^{\mathrm{T}} \nabla^2 g(x^k, y^k) \begin{pmatrix} x - x^k \\ y - y^k \end{pmatrix} \leq 0 \quad \forall k \in K \\
&\quad Ax + Ey \leq b \\
&\quad \eta \in \mathbb{R}^1 \\
&\quad x \in X \subset \mathbb{R}^n \\
&\quad y \in Y \subset \mathbb{Z}^p
\end{aligned}$$

$$(M^k_{OA-QCUT})$$

We add the quadratic cuts of the continuous and discrete variables in the M^k_{QOA} master problem.

Figure 4 provides the geometric illustration of the nonlinear objective function and feasible region with scaled quadratic cuts. This objective function and feasible region represent the following mathematical programming problem involving convex functions,

$$\begin{aligned}
&\min \quad f = x_1^4 + 3 \\
&s.t \quad g_1 : 20 - x_1^2 - x_2 \geq 0 \\
&\quad g_2 : 5 + 2(x_1 - 1)^2 - x_2 \leq 0 \\
&\quad g_3 : 15\sqrt{x_1} - x_2 \leq 0 \\
&\quad x_1 \leq 3 \\
&\quad x_2 \leq 30 \\
&\quad (x_1, x_2) \in \mathbb{R}^2_+
\end{aligned} \tag{22}$$

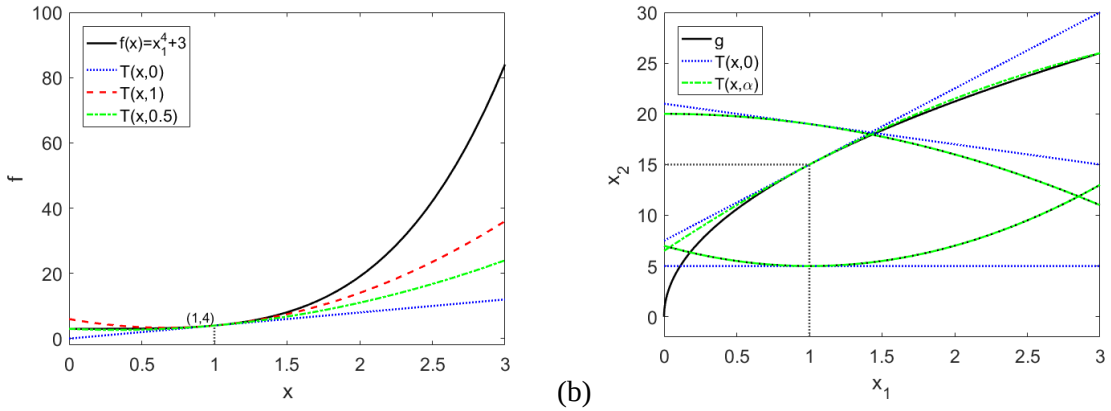
The OA relaxation of the problem in Eq. (22) generated at the point $x_0 = (1,15)$ can be written as follows,

$$\begin{aligned} \min \quad & T(x,0) = 4x_1 \\ s.t \quad & T_1(x,0): 21 - 2x_1 - x_2 \geq 0 \\ & T_2(x,0): 5 - x_2 \leq 0 \\ & T_3(x,0): \frac{15}{2} + \frac{15}{2}x_1 - x_2 \leq 0 \\ & x_1 \leq 3 \\ & x_2 \leq 30 \\ & (x_1, x_2) \in \mathbb{R}^2_+ \end{aligned} \tag{23}$$

In Figure 4(a), we can see the scaled quadratic cut with $\alpha = 0.5$ is the underestimation of the nonlinear function. Figure 4(b) is the illustration of the scaled quadratic cuts for each of the nonlinear constraints. The scaled quadratic problem is as follows,

$$\begin{aligned} \min \quad & T(x,0.5) = 4x_1 + 0.5\frac{12}{2}(x_1 - 1)^2 \\ s.t \quad & T_1(x,1): 21 - 2x_1 - x_2 - (x_1 - 2)^2 \geq 0 \\ & T_2(x,1): 5 - x_2 + 2(x_1 - 1)^2 \leq 0 \\ & T_3(x,0.536): \frac{15}{2} + \frac{15}{2}x_1 - x_2 + 0.536\frac{15}{8}(x_1 - 2)^2 \leq 0 \\ & x_1 \leq 3 \\ & x_2 \leq 30 \\ & (x_1, x_2) \in \mathbb{R}^2_+ \end{aligned} \tag{24}$$

Note that for every nonlinear constraint, the scalar α is calculated via Eq. (20), and therefore a different value is obtained for each constraint. For the quadratic constraints g_1 and g_2 the value is one, and for the third constraint it is 0.536. The scaled quadratic cuts underestimate the objective function, and overestimate the convex feasible region. As the second order expansion is closer to the nonlinear function, the lower bound of the master problem with quadratic cuts is tighter than one with linear cuts.



(a) (b)

Figure 4. Geometrical representation of scaled quadratic cuts of the master problem of OA regarding the (a) objective function and (b) constraints defining the feasible region

With scaled quadratic cuts, the master problem of OA becomes a MIQCP with linear objective function and quadratic constraints. Since the Hessian matrices of functions f and g are positive semi-definite in all points (x^k, y^k), the master problem MIQCP of OA is convex.

Proposition 3. The convex MINLP problem P and the following master problem $MIQCP$ have the same optimal solution (x^*, y^*),

$$Z^k_{LB,OA-QCUT} = \min_{\eta,x,y} \quad \eta$$

$$s.t \quad \eta \geq f(x^k, y^k) + \nabla f(x^k, y^k)^T \begin{pmatrix} x - x^k \\ y - y^k \end{pmatrix} + \frac{\alpha^k}{2} \begin{pmatrix} x - x^k \\ y - y^k \end{pmatrix}^{\mathrm{T}} \nabla^2 f(x^k, y^k) \begin{pmatrix} x - x^k \\ y - y^k \end{pmatrix} \quad \forall k \in K^*$$

$$g(x^k, y^k) + \nabla g(x^k, y^k)^T \begin{pmatrix} x - x^k \\ y - y^k \end{pmatrix} + \frac{\beta^k}{2} \begin{pmatrix} x - x^k \\ y - y^k \end{pmatrix}^{\mathrm{T}} \nabla^2 g(x^k, y^k) \begin{pmatrix} x - x^k \\ y - y^k \end{pmatrix} \leq 0 \quad \forall k \in K^*$$

$$Ax + Ey \leq b$$

$$\eta \in \mathbb{R}^1$$

$$x \in X \subset \mathbb{R}^n$$

$$y \in Y \subset \mathbb{Z}^p$$

$$(MIQCP)$$

where K^* is the set of optimal solutions to the problems $NLP(y^k)$ for all y^k, and which are feasible for the original problem P.

Proof. Since the generated scaled cuts do not cutoff the feasible solutions of convex nonlinear problem (Proposition 2) and the optimal theorem of OA (Duran and Grossmann, 1986; Fletcher and

Leyffer, 1994), we can trivially prove Proposition 3.

We usually solve a relaxation of MIQCP, which is $M^k_{OA-QCUT}$ with $k \in K$, and K is a subset of K^*. The solution of the MIQCP relaxation corresponds to a lower bound of the optimal solution of P.

Since the scaled quadratic cuts in $M^k_{OA-QCUT}$ are accumulated as the OA main iterations proceeds, the master problem can be solved to obtain a non-decreasing sequence of lower bounds. Therefore, we develop an expansion of OA with scaled quadratic cuts, OA-QCUT. Note that if the original MINLP problem P has quadratic objective and/or constraints, the problem $MIQCP$ will be exactly as the original problem, and therefore will be solved by solving the master problem.

OA-QCUT method is also a decomposition method for convex MINLP problems, whose difference with the general OA is that the master problem is a MIQCP. The subproblem of OA-QCUT is also the NLP with fixed discrete variables of P. Making use of the solution of $NLP(y^k)$, we can obtain the accumulated scaled quadratic cuts for master problem, and update the upper bound of problem P. The solution of the master problem MIQCP then supplies the solution of discrete variables and updates the lower bounds of P.

5. Improved MINLP methods with quadratic cuts

In Su et al. (2015), effective computational strategies for MINLP methods are presented including multi-generation cuts, hybrid cuts for OA, GBD and PSC. Several new improved MINLP methods are presented next for convex MINLP problems based on the integration of the scaled quadratic cuts with the strategies of multi-generation cuts and PSC.

5.1. Multi-generation quadratic cuts for OA (OA-MQCUT)

The OA method with scaled quadratic cuts inherits the computational difficulties of the master problem in the original OA method regarding the accumulation of generated cuts, which can be even more challenging given that the problem now is an MIQCP. The strategy of multi-generation cuts for OA aims to decrease the computational effort of the MILP master problem by merging solution of multiple MILP problems. Therefore, we try to integrate multi-generation cuts (MCUT) with scaled quadratic cuts for OA, which generates multiple quadratic cuts by parallel solutions of NLP subproblems in one main iteration. Suppose the number of multi-generation cuts is $|S|$. The equations of multi-generation scaled quadratic cuts are as follows,

$$\left.\begin{array}{l} \eta \geq f(x^{k,s},y^{k,s})+\nabla f(x^{k,s},y^{k,s})^{T}\begin{pmatrix} x-x^{k,s} \\ y-y^{k,s}\end{pmatrix}+\dfrac{\alpha^{k,s}}{2}\begin{pmatrix} x-x^{k,s} \\ y-y^{k,s}\end{pmatrix}^{\mathrm{T}}\nabla^{2} f(x^{k,s},y^{k,s})\begin{pmatrix} x-x^{k,s} \\ y-y^{k,s}\end{pmatrix} \\ g(x^{k,s},y^{k,s})+\nabla g(x^{k,s},y^{k,s})^{T}\begin{pmatrix} x-x^{k,s} \\ y-y^{k,s}\end{pmatrix}+\dfrac{\beta^{k,s}}{2}\begin{pmatrix} x-x^{k,s} \\ y-y^{k,s}\end{pmatrix}^{\mathrm{T}}\nabla^{2} g(x^{k,s},y^{k,s})\begin{pmatrix} x-x^{k,s} \\ y-y^{k,s}\end{pmatrix}\leq 0 \end{array}\right\} \quad \forall k,s\in S \tag{25}$$

Here, we can simplify the calculation of the scaling coefficients. Assume at the k^{th} iteration of OA, we obtain the scaled vector $\left(\alpha^{k,s},\beta^{k,s}\right)$. The initial scaled coefficient of the $k+1^{th}$ iteration is given by,

$$\begin{aligned} \alpha^{k+1} &= \min_{s}\left\{\alpha^{k,s}\right\} \quad \forall k \\ \beta^{k+1} &= \min_{s}\left\{\beta^{k,s}\right\} \quad \forall k \end{aligned} \tag{26}$$

However, we need to verify whether the scaled cuts are valid before using $\left(\alpha^{k+1},\beta^{k+1}\right)$. If all these generated cuts are valid, $\left(\alpha^{k+1},\beta^{k+1}\right)$ can be used at the next iteration. Otherwise, we need to recalculate the scaled vector according to Proposition 2. Therefore, the scaled array $\left(\alpha^{k+1},\beta^{k+1}\right)$ yields a monotonic decreasing sequence.

The main steps of multi-generation quadratic cuts are shown in Figure 5, and are as follows:

OA-MQCUT algorithm:

Initialization: set $k = 0, UBD = \infty$, specify |S|

(1) Solve the relaxed MINLP with continuous integer variables. Let the solution be $\left(x^k, y^k\right)$.

(2) Derive the scaled quadratic cuts at $\left(x^k, y^k\right)$ as in Eq (21), and add to the current relaxation M^k of the master problem. Let the multiple solutions to the master problem be $\left(x^{k+1,s}, y^{k+1,s}\right), s \in S$.

Repeat

(3) Solve the subproblem $NLP(y^{k,s})$, or the feasibility problem $F(y^{k,s})$ if $NLP(y^{k,s})$ is infeasible, and let the solutions be $x^{k,s}$.

(4) If the $NLP(y^{k,s})$ is feasible, let x^* be the optimal solution to the problem and if $f^{k,s} < UBD$, then update current best solution by setting $x^* = x^{k,s}, y^* = y^{k,s}, UBD = f^{k,s}$.

(5) Update the scaled vectors $\alpha^{k,s}$ and $\beta^{k,s}$ as in Eq. (20) and (26).

(6) Derive the scaled quadratic cuts at $\left(x^k, y^k\right)$ as in Eq. (25), and add to the current relaxation M^k of the master problem.

(7) Solve the current M^k, getting a set of integer assignment y^{k+1}. Set $k = k + 1$.

Continue until the termination conditions are satisfied.

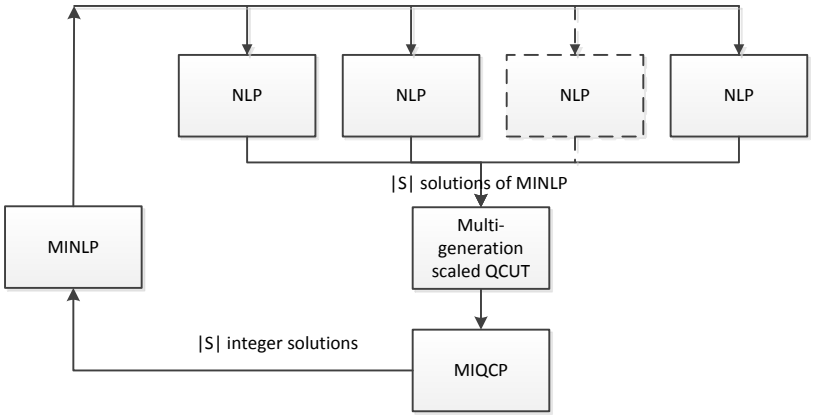


Figure 5. Main steps of multi-generation quadratic cuts of OA (OA-MQCUT)

5.2. Hybrid linear and quadratic cuts for OA (OA-HCUT) with multi-generation strategy (OA-MHCUT)

Considering the computational complexity of MIQCP, we develop hybrid linear and scaled quadratic cuts (HCUT) for OA based on a multi-generation strategy. We separate the OA solution process into two stages by iteration number. The main steps are as follows.

Stage 1. In first stage, multi-generation linear supporting cuts are constructed and accumulated into the MILP master problem, which is MCUT-OA (Su et al., 2015) until the specified limit of iterations is reached.

Stage 2. In the second stage, multi-generation scaled quadratic cuts are constructed and accumulated in the master problem, which changes the master problem from MILP into MIQCP. This is also OA-MQCUT proposed in Section 5.1. The solution process terminates when the termination conditions of the OA method are satisfied.

Since the scaled quadratic cuts are tighter than the linear supporting cuts, the OA with multi-generation hybrid cuts should accelerate convergence in the second stage. Also with the process of accumulating generating cuts, the lower bounds from the master problem are non-decreasing.

We also can extend the hybrid cuts strategy, like generating quadratic cuts in the first stage, then

generating linear cuts in the second stage. Moreover, we can divide the solution process into multiple stages, and generate different kinds of cuts in a specified certain stage.

5.3. Partial surrogate quadratic and multi-generation cuts

Partial surrogate cuts are one type of cuts between OA cuts and GBD cuts, which are derived from projecting on the nonlinear variables and applying KKT conditions (Quesada and Grossmann, 1992; Su et al., 2015). Since the PSC includes continuous nonlinear variables, we can extend it to scaled quadratic cuts as follows.

$$\eta \geq f\left(v^k, w, y\right) + \begin{pmatrix} \lambda^k \\ -\mu^k \end{pmatrix}^T \begin{bmatrix} g\left(v^k, w, y\right) & 0 \\ 0 & A_2 \end{bmatrix} \begin{pmatrix} 1 \\ v - v^k \end{pmatrix} + \frac{\alpha^k}{2}\left(v - v^k\right)^T \nabla^2 L^k \left(v - v^k\right) \quad \forall k \tag{27}$$

where L^k is the projection of the Lagrange function in the nonlinear continuous variable *v* space. Eq. (27) can be derived from the procedure of PSC cuts with the 2nd-order Taylor expansion of nonlinear functions (Quesada and Grossmann, 1992).

The scalars are computed in an equivalent manner as in Eq. (20), as follows.

$$\frac{\alpha^k}{2} = \min_{\left(v^v, w^v, y^v\right) \in V} \left\{ \frac{f(v^v, w^v, y^v) - f(v^k, w^v, y^v) - \lambda g(v^k, w^v, y^v) + \mu A_2 (v^v - v^k)}{(v^v - v^k)^T \nabla^2 L(v^k, w^k, y^k)(v^v - v^k)} \right\}$$
$$\forall k, v^v \neq v^k \tag{28}$$

Eqs. (27) and (28) define the Partial Surrogate Quadratic Cuts (PSQC), which are an underestimation for convex nonlinear functions based on Proposition 2. Furthermore, if the calculated scalar α based on Eq. (28) is greater than 1, we set the scalar α as 1 given the fact that in that case we do not want to overestimate the 2nd order Taylor approximation.

We develop the PSC-QCUT method based on the PSC method using scaled quadratic cuts as in Eq. (27). Compared to the OA cuts, PSC cuts are not as tight for the relaxation of nonlinear constraints

(Su et al., 2015). The advantage of PSC is generating fewer cuts than OA for an iteration. These advantages generally are inherited from PSC to PSC-QCUT.

We integrate the multi-generation strategy with PSQC, leading to the Partial Surrogate Multi-generation Quadratic Cuts (PSC-MQCUT) generating multiple cuts in a single iteration as follows.

$$\begin{aligned}\eta \geq f\left(v^{k,s},w,y\right)+\begin{pmatrix}\lambda^{k,s}\\-\mu^{k,s}\end{pmatrix}^{T}\begin{bmatrix}g\left(v^{k,s},w,y\right) & 0\\0 & A_2\end{bmatrix}\begin{pmatrix}1\\v-v^{k,s}\end{pmatrix}\\+\frac{\alpha^{k,s}}{2}\left(v-v^{k,s}\right)^{T}\nabla^2 L^{k,s}\left(v-v^{k,s}\right)\quad\forall k,s\in S\end{aligned} \tag{29}$$

As is known, the lower bounds from PSC are not tighter than the lower bounds from OA. The same is true for PSC-QCUT and OA-QCUT. However, the number of generated cuts in the PSC method in one iteration are fewer than the number of OA cuts, which may reduce the computational time of solving the master problem MIQCP.

5.4. Proposed MINLP algorithms

In summary, we develop 6 improved MINLP methods, OA-QCUT, OA-MQCUT, OA-HCUT, OA-MHCUT, PSC-QCUT and PSC-MQCUT integrating the proposed scaled quadratic cuts with multigeneration linear cuts for OA and PSC. The relationship of the improved algorithms with original OA and PSC is illustrated in Figure 6.

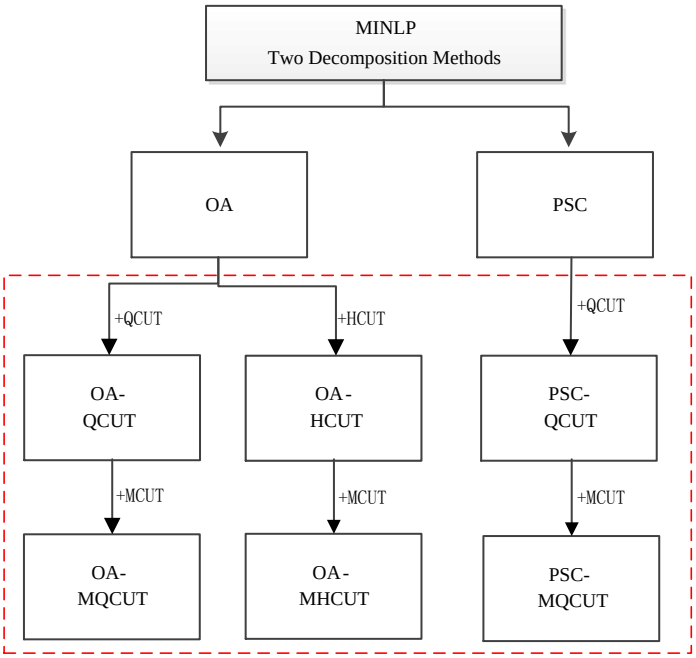


Figure 6. Relationships of OA-QCUT, OA-MQCUT, OA-HCUT, OA-MHCUT, PSC-QCUT and PSC-MQCUT algorithms with OA and PSC

The main steps of the *OA-QCUT/PSC-QCUT* algorithms are as follows.

OA-QCUT/PSC-QCUT algorithms:

Initialization: set $k=0, UBD=\infty$

Solve the relaxed MINLP with continuous integer variables. Let the solution be $\left(x^k, y^k\right)$.

(1) Compute the initial scaled vector α^k based on Eq. (20)/(28).

(2) Expand the nonlinear functions on $\left(x^k, y^k\right)$ as in Eq. (21)/(27) for the MINLP. Solve the integer relaxed MIQCP. Let the optimal solution be $\left(x^{k+1}, y^{k+1}\right)$.

Repeat

(3) Solve the subproblem $NLP(y^k)$, or the feasibility problem $F(y^k)$ if $NLP(y^k)$ is infeasible, and let the solutions be x^k.

(4) If the $NLP(y^k)$ is feasible, let x^* be the optimal solution to the problem and if $f^k < UBD$, then update current best solution by setting $x^* = x^k, y^* = y^k, UBD = f^k$.

(5) Update the scaled vector α^k as in Eq. (20)/(28).

(6) Derive the scaled quadratic cuts at $\left(x^k, y^k\right)$ as in Eq (21)/(27), and add to the current relaxation M^k of the master problem.

(7) Solve the current M^k, getting a set of integer assignment y^{k+1}. Set $k = k+1$.

Continue until the termination conditions are satisfied.

The main steps of OA-HCUT are divided into two stages, in which the first stage is the general OA procedure, and the second stage is the OA-QCUT procedure as above. Furthermore, the main steps of OA-MQCUT, OA-MHCUT, PSC-MQCUT are added to the solution process of each subproblem $NLP(y^{k,s})$ in parallel for one iteration of OA-QCUT/OA-HCUT/PSC-CUT.

6. Numerical experiments

We first discuss the solution methods of MIQCP master problems, and solve 44 MINLP benchmark problems with different problem size and number of nonlinear functions using the proposed methods, OA-QCUT, OA-MQCUT, OA-HCUT, OA-MHCUT, PSC-QCUT, PSC-MQCUT. The computational performance is compared with OA and PSC, and the MINLP solvers DICOPT, BONMIN, SBB and SCIP.

For the MINLP problems with discrete variables in nonlinear functions, we have two alternative ways: one is expanding the nonlinear terms with discrete variables to 1st order, and the other is introducing new equivalent continuous variables instead of the discrete variables, and expanding the new variables to 2nd order as the other continuous variables.

We implement the benchmark MINLP test problems in GAMS 24.8.2 (Bussieck and Meeraus, 2004). The operating system is Windows 7, with an Intel Core 2 Duo processor, a CPU of 3.4 GHz and 16 GB of RAM. The solutions of the NLP problems are obtained using the solvers CONOPT 3.17C

(Drud, 1985) and IPOPT 3.12 (Wächter and Biegler, 2005), and the solutions of the MILP/MIQCP problems are obtained using CPLEX 12.7.0.0 (IBM Corp. and IBM, 2009). Three stopping criteria for the proposed methods and solvers are used. The first one is if the objective of the last MIP master problem is greater than the best NLP solution found, namely a 'crossover' occurred; the second one is the maximum major iterations; and the third one is the maximum solution CPU time. The major iteration limit is 500, and the maximum CPU solution time is 3600 seconds. The CPU solution time only includes the time used solving the master problem and the subproblems, and does not include the time calculating the values of the scale parameters α and β. These parameters were calculated using Eq. (20), which resulted in the same values as solving the problem stated in Eq. (14). None of the methods to had a considerable time compared to the overall MILP and NLP solution times.

The MINLP test problems are classified into three groups: 7 simple problems, 15 problems with special structure, and 22 medium/large scale problems (Trespalacios and Grossmann, 2014).

6.1. Simple MINLP problems

The 7 simple MINLP problems are tested taken from the MINLP Lib[1]. Five problems were partly also used in Duran and Grossmann (1986). In the problems Synthes1- Synthes3, the nonlinear terms are the exponential or logarithmic functions of only continuous variables, which are separable from the discrete variables. Problem Gkocis has a nonlinear objective function, and the problem Alan has only one nonlinear constraint. The problem Ex1223b has discrete variables involved in the nonlinear functions, while the problem St_e14 is an equivalent transformation of the Ex1223b problem. Here, we expand the nonlinear terms of Ex1223b with discrete variables to first order, and all nonlinear terms to second order using the Taylor expansion.

[1] http://www.gamsworld.org/minlp/minlplib.htm accessed in January 2017

The number of variables and constraints for the 7 problems are shown in the Appendix. The computational results of DICOPT, and implementations of OA, OA-QCUT and OA-MQCUT are listed in Table 1. The initial scalar vector α is computed based on Eq. (20), and updated by Eq. (26). We find that the scalars α are near 0.5 for the first 3 problems, and almost 1 for the last 4 problems.

All solution methods were able to obtain the optimal solutions for the 7 problems. For the simpler problems, fewer iterations and shorter CPU times were required by OA-QCUT and OA-MQCUT than OA with linear cuts. OA-QCUT requires fewer iterations than DICOPT, although it generally requires slightly longer CPU times.

Table 1. Computational results of DICOPT, OA, OA-QCUT and OA-MQCUT for small MINLP problems

Problems	DICOPT		OA[a]		OA-QCUT		OA-MQCUT		α_{min}
	Iter.	CPU(s)	Iter.	CPU(s)	Iter.	CPU(s)	Iter.	CPU(s)	
Synthes1	4	0.8	5	2.0	3	1.5	3	1.3	0.41
Synthes2	7	0.5	3	0.9	3	0.8	3	1.1	0.46
Synthes3	11	0.6	7	3.0	3	1.2	2	0.8	0.58
Gkocis	3	0.4	3	1.2	2	0.7	2	0.8	1.0
Alan	5	0.4	5	2.6	3	1.1	2	0.9	1.0
Ex1223b	7	0.4	4	4.5	3	1.0	3	1.0	1.0
St_e14	6	0.5	6	5.3	2	0.8	2	0.7	0.99

[a] It is GAMS implementation of OA, which is same for the following tables.

[b] It is the value of the scalar value for α.

For the same problem set, the computational results of OA with hybrid cuts (OA-HCUT) and multi-generation hybrid cuts (OA-MHCUT), and GAMS implementations of PSC, PSC-QCUT, PSC-MQCUT are shown in Table 2. For OA-HCUT, we obtain linear cuts in the first iteration, and then quadratic cuts in the following iterations.

Although the scaled quadratic cuts are generated at the second iteration for the OA-HCUT method, OA-HCUT requires either fewer iterations or shorter solution times than OA for all problems in this

comparison. OA-MHCUT was even more efficient than OA-HCUT, except for the Synthes3 and St_e14 problems.

PSC-QCUT performs better than PSC, with fewer iterations and smaller CPU times on 75% of a total of 7 problems. PSC-MQCUT generally requires fewer iterations and shorter CPU times than PSC-QCUT in the simple MINLP instances, except for the Gkocis problem.

Table 2. Computational results of OA-HCUT, OA-MHCUT, PSC, PSC-QCUT, PSC-MQCUT for small MINLPs

Problems	OA-HCUT		OA-MHCUT		PSC		PSC-QCUT		PSC-MQCUT		α_{min} [a]
	Iter.	CPU(s)	Iter.	CPU(s)	Iter.	CPU(s)	Iter.	CPU(s)	Iter.	CPU(s)	
Synthes1	3	1.0	2	0.9	4	2.2	3	1.7	3	1.3	1
Synthes2	3	1.0	2	0.8	6	2.8	5	3.0	3	1.3	1
Synthes3	3	1.3	4	2.0	12	6.4	5	2.5	2	1.0	1
Gkocis	2	1.1	2	1.0	3	1.3	3	1.6	3	1.7	1
Alan	3	1.0	2	0.8	5	2.7	4	2.2	3	2.1	1
Ex1223b	4	1.4	3	1.5	6	2.8	6	3.2	3	1.5	1
St_e14	2	1.0	2	1.1	6	3.0	4	2.9	3	1.9	1

[a]It is the scalar values of α for PSC-QCUT and PSC-MQCUT.

6.2. MINLP with special structure

The set of Constrained Layout (CLAY) and Strip Layout (SLAY) problems[2] is special, since all the MINLPs in it have quadratic nonlinear constraints. The chosen instances were the ones that replaced the disjunctions using the Big-M method (Trespalacios and Grossmann, 2014). Since the constraints are quadratic, the scalar vector α is set as 1. The computational results for these instances are shown in Table 3.

Since the quadratic cut is equal to the quadratic function in this problem set, OA-QCUT and

[2] http://egon.cheme.cmu.edu/ibm/page.htm accessed in January 2017

OA-MQCUT terminate at the first normal main iteration after the initial one. As expected, OA and DICOPT require more iterations and longer CPU times to obtain the optimal solutions. OA-QCUT performs better than OA-MQCUT on CPU times for 88.6% problems of the CLAY and SLAY testset. More CPU time is needed to generate multiple cuts and solve the MIQCP master problems using OA-MQCUT. We also solve the CLAY and SLAY instances as MIQCP directly using CPLEX, which results with smallest CPU times among all methods.

Table 3. Computational results of DICOPT, OA, OA-QCUT, OA-MQCUT for CLAY and SLAY instances from (Trespalacios and Grossmann, 2014)

Problems	CPLEX	DICOPT[a]		OA		OA-QCUT		OA-MQCUT	
	CPU(s)	Iter.	CPU(s)	Iter.	CPU(s)	Iter.	CPU(s)	Iter.	CPU(s)
Clay42	0.8	10	3.2	10	7.8	2	2.9	2	4.6
Clay43	2.8	12	4.3	9	6.3	2	5.4	2	13.4
Clay44	2.2	20	14.2	17	17.6	2	10.5	2	19.7
Clay45	16.5	25	36.7	19	62.4	2	33.4	2	60.5
Clay52	4.8	10	7.4	12	18.5	2	9.7	2	11.8
Clay53	5.7	11	6.9	7	7.6	2	11.0	2	33.5
Clay54	17.4	21	101.4	19	216.5	2	50.2	2	56.5
Clay55	58.0	40	498.3	33	601.8	2	115.2	2	207.6
Slay04	0.4	9	0.8	10	4.8	2	0.7	2	0.9
Slay05	0.2	26	1.7	11	3.7	2	1.1	2	1.2
Slay06	0.3	50	8.4	20	19.3	2	4.9	2	2.5
Slay07	0.3	500	-	24	17.1	2	5.5	2	1.5
Slay08	0.4	500	-	25	30.2	2	9.8	2	4.2
Slay09	0.4	79	94.4	39	448.7	3	22.1	2	17.3
Slay10	1.7	500	-	51	2433.8	3	8.7	2	5.6

[a]IPOPT used as NLP solver for DICOPT.

The same instances were solved using the OA-HCUT, OA-MHCUT, PSC, PSC-QCUT, and PSC-MQCUT methods. The computational results of this comparison are shown in Table 4. Note that

we obtain the OA cuts for the first master problems of PSCs and the following iteration when the NLP subproblem is infeasible.

Although the iteration numbers of OA-HCUT and OA-MHCUT are the same in Table 4, the CPU times of OA-MHCUT are generally longer than OA-HCUT except for the problem Slay08-10. Because of the structure of the CLAY and SLAY instances, the master problems of OA-MHCUT generates more quadratic cuts than the master problems of OA-HCUT. OA-HCUT requires longer solution time than OA-QCUT for 90.9% problems in this section.

Table 4. Computational results of OA-HCUT, OA-MHCUT, PSC, PSC-QCUT and PSC-MQCUT for CLAY and SLAY instances (Trespalacios and Grossmann, 2014)

Problems	OA-HCUT		OA-MHCUT		PSC[a]		PSC-QCUT		PSC-MQCUT	
	Iter.	CPU(s)	Iter.	CPU(s)	Iter.	CPU(s)	Iter.	CPU(s)	Iter.	CPU(s)
Clay42	3	5.9	3	8.8	8	4.3	3	2.5	3	3.3
Clay43	3	15.0	3	14.6	9	4.6	3	5.0	3	7.2
Clay44	3	20.4	3	70.1	13	8.8	3	14.9	3	23.7
Clay45	3	91.7	3	144.7	17	16.9	3	43.3	3	78.5
Clay52	3	16.6	3	41.5	8	8.9	3	8.7	3	12.3
Clay53	3	20.7	3	134.1	8	9.1	3	8.6	3	27.9
Clay54	3	115.5	3	308.6	12	79.0	3	57.2	3	135.2
Clay55	3	225.8	3	421.2	28	640.9	3	169.9	3	348.8
Slay04	3	1.8	3	3.1	3	2.0	2	0.9	2	1.1
Slay05	3	2.1	3	2.9	3	3.0	2	1.5	2	1.9
Slay06	3	2.4	3	3.3	3	11.2	2	1.9	2	1.7
Slay07	3	2.2	3	2.3	3	102.0	3	3.1	2	3.0
Slay08	3	12.0	3	9.0	3	1003.1	2	2.7	2	3.8
Slay09	3	5.6	3	9.2	3	1019.2	2	2.9	2	44.1
Slay10	3	1005.5	3	534.9	3	1006.6	2	16.6	2	4.8

[a]IPOPT used as NLP solver for PSC.

There are infeasible subproblems for the CLAY and SLAY instances, which generate the infeasible

cuts in the PSC-QCUT method. These cuts are similar to those generated by OA-QCUT. This fact allows the PSC-QCUT and PSC-MQCUT to solve the compared instances in only 3 or 2 iterations. Since PSC-MQCUT generated more scaled quadratic cuts in the master problems than PSC-QCUT, PSC-MQCUT requires longer solution times than PSC-QCUT, except for 3 SLAY instances. The computational results in Table 3 show that OA-QCUT decrease greatly reduces the number of iterations compared with DICOPT, and OA. This is an expected behavior since the nonlinear constraints of these instances are quadratic, which would mean that a quadratic approximation would be exact.

The nonlinear constraints in the CLAY instances ensured that the feasible region between two variables was a circle. One specific case was a circle of radius 6 and centered in the point $(x_1, x_2) = (15,10)$. Expressing the constraints with respect to one of the variables, we obtain the following.

$$\begin{aligned} g_1 &: x_2 \le 10 + \sqrt{(x_1 - 15)^2 - 6^2} \\ g_2 &: x_2 \ge 10 - \sqrt{(x_1 - 15)^2 - 6^2} \end{aligned} \tag{30}$$

Assuming that we have expansion points at $x_0 = 11$ and $x_0 = 20$, the linear supporting planes derived from the 1st order Taylor approximation will be as follows.

$$\begin{aligned} T_{1,1}(x,0) &: x_2 \le 10 + \sqrt{20} + \frac{\sqrt{20}(x_1 - 11)}{5} \\ T_{2,1}(x,0) &: x_2 \ge 10 - \sqrt{20} - \frac{\sqrt{20}(x_1 - 11)}{5} \\ T_{1,2}(x,0) &: x_2 \le 10 + \sqrt{11} - \frac{5\sqrt{11}(x_1 - 20)}{11} \\ T_{2,2}(x,0) &: x_2 \ge 10 - \sqrt{11} + \frac{5\sqrt{11}(x_1 - 20)}{11} \end{aligned} \tag{31}$$

On the other hand, after determining the minimum α for every constraint such that every cut is a valid underestimator we obtain the following quadratic cuts.

$$
\begin{aligned}
&T_{1,1}(x,0.556): x_2 \le 10+\sqrt{20}+\frac{\sqrt{20}(x_1-11)}{5}-\frac{0.556}{2}\frac{9\sqrt{20}(x_1-11)^2}{100}\\
&T_{2,1}(x,0.556): x_2 \ge 10-\sqrt{20}-\frac{\sqrt{20}(x_1-11)}{5}+\frac{0.556}{2}\frac{9\sqrt{20}(x_1-11)^2}{100}\\
&T_{1,2}(x,0.394): x_2 \le 10+\sqrt{11}-\frac{5\sqrt{11}(x_1-20)}{11}-\frac{0.394}{2}\frac{36\sqrt{11}(x_1-20)^2}{121}\\
&T_{2,2}(x,0.394): x_2 \ge 10-\sqrt{11}+\frac{5\sqrt{11}(x_1-20)}{11}+\frac{0.394}{2}\frac{36\sqrt{11}(x_1-20)^2}{121}
\end{aligned}
\tag{32}
$$

The feasible region of one of the CLAY instance's nonlinear constraints is illustrated in Figure 7.

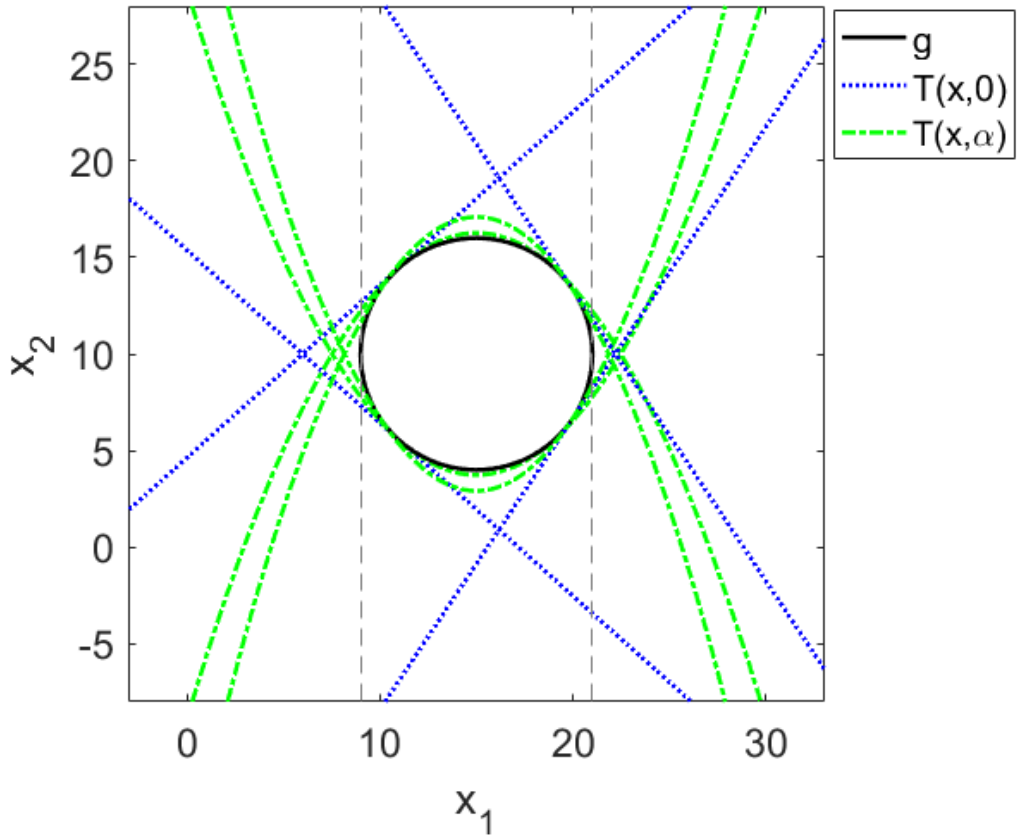


Figure 7. Illustration of feasible region of a CLAY instance with linear cuts and quadratic cuts

From the Figure 7 and compared with the linear approximation of the feasible region of CLAY, the quadratic approximation of the feasible region is tighter. That explains why OA-QCUT can perform better than the original OA.

6.3. Medium /large scale MINLP problems

22 MINLP problems with medium/large scale are tested in this subsection[3](Trespalacios and Grossmann, 2014). Aiming at the general MINLP with all kinds of nonlinear functions, we consider to expand the nonlinear terms satisfying Positive Semi-Definitiveness (PSD) to second order. Based on Eq. (20), the value of α is computed for each problem in order to guarantee the optimal solutions. For the Batch instances, we constructed the first master problems of PSC, PSC-QCUT and PSC-MQCUT using PSC and PSC-QCUT cuts, respectively. For other problems in this subsection, we used the OA cuts for the first master problems of PSC, PSC-QCUT and PSC-MQCUT.

The computational results of DICOPT, OA, OA-QCUT and OA-MQCUT are shown in Table 5, while the computational results of OA-HCUT, OA-MHCUT, PSC, PSC-QCUT and PSC-MQCUT for these instances are shown in Table 6.

Table 5. Computational results of DICOPT, OA, OA-QCUT and OA-MQCUT for medium/large scale MINLPs

Problems	DICOPT		OA		OA-QCUT		OA-MQCUT		α_{min}
	Iter.	CPU(s)	Iter.	CPU(s)	Iter.	CPU(s)	Iter.	CPU(s)	
Batch03	2	0.4	2	0.6	2	0.6	2	0.7	1
Batch06	4	0.3	5	1.7	3	1.3	3	2.2	1
Batch08	5	0.5	5	2.0	4	1.4	4	2.7	0.85
Batch10	10	4.7	12	6.3	2	7.7	2	11.2	1
Batch12	5	4.2	6	9.3	2	13.3	2	17.7	1
Batch15	6	6.5	7	12.3	3	42.2	3	46.6	1
Batch20	4	6.2	10	30.4	2	33.8	2	76.1	1
Csched1a	2	0.3	4	1.2	2	0.6	2	0.7	1
Csched1	20	1.2	8	2.3	2	1.1	2	0.8	1

[3] http://egon.cheme.cmu.edu/ibm/page.htm

Csched2a	90	16.5	55	34.1	3	1.8	3	2.1	1
Csched2	160	146.5	117	254.3	4	2.7	3	3.1	1
Flay02	4	0.4	3	1.0	3	0.9	2	0.6	1
Flay03	34	1.3	9	3.7	2	1.0	2	1.0	1
Flay04	300	100.9	18	15.8	3	3.0	3	4.2	1
Flay05	500	-	25	306.0	2	4.2	2	4.6	1
Flay06	-	3600	-	3600	2	34.7	2	68.1	1
Proc_21a	25	4.0	8	4.3	4	9.2	3	15.0	1
Proc_21b	25	3.6	6	3.0	4	4.2	3	5.0	1
Proc_31a	30	4.6	5	2.2	3	2.6	2	2.9	1
Proc_31b	28	4.5	6	4.1	2	2.3	2	2.9	1
Proc_36a	32	8.3	6	4.2	3	5.4	3	9.5	1
Proc_36b	35	8.7	6	5.1	3	5.7	3	8.6	1

Note that OA-QCUT and OA-MQCUT requires only 2 or 4 iterations, although the CPU solution times are not shorter than DICOPT for Batch problems. This trend is reversed in for the Csched problems except Csched1a. DICOPT failed to solve 2 of the Farm Layout (Flays), and OA fails on 1 Flay instance, but OA-QCUT and OA-MQCUT solved all Flay instances. In all the instances except from Batch08, where the value of the $\alpha_{\min}$ is equal to 0.85, $\alpha_{\min}$ is equal to 1. This means that the nonlinear functions were underestimated by the second order Taylor expansion for this set of test problems.

Table 6. Computational results of OA-HCUT, OA-MHCUT, PSC, PSC-QCUT, PSC-MQCUT for Medium/large scale MINLPs

Problems	OA-HCUT		OA-MHCUT		PSC		PSC-QCUT		PSC-MQCUT		$\alpha_{\min}$ [a]
	Iter.	CPU(s)	Iter.	CPU(s)	Iter.	CPU(s)	Iter.	CPU(s)	Iter.	CPU(s)	

Batch03	2	0.8	2	0.8	14	5.4	9	6.7	8	8.2	1
Batch06	3	7.2	2	1.3	280	264.7	47	287.8	18	63.8	1
Batch08	4	1.4	4	2.4	2	0.6	4	5.4	2	1.4	1
Batch10	3	10.6	2	6.1	4	1.4	3	8.2	3	6.7	1
Batch12	3	20.5	4	36.8	4	2.0	3	27.6	3	15.2	1
Batch15	3	26.7	2	18.8	3	1.2	3	1.5	3	1.8	1
Batch20	3	42.4	3	56.4	3	1.3	3	1.6	3	2.0	1
csched1a	3	1.0	2	0.8	4	1.2	2	0.7	2	0.8	1
csched1	5	3.2	2	0.8	8	2.3	2	0.7	2	0.8	1
csched2a	4	1.9	3	3.2	100	95.2	3	1.8	3	2.6	1
csched2	4	2.0	3	3.9	117	217.4	3	1.9	3	2.2	1
Flay02	3	1.1	3	1.4	500	-	500	-	500	-	1
Flay03	3	1.4	3	2.2	500	-	500	-	500	-	1
Flay04	3	2.6	3	3.0	500	-	500	-	500	-	1
Flay05	4	5.8	2	3.5	500	-	500	-	500	-	1
Flay06	3	19.3	3	27.8	500	-	500	-	500	-	1
Proc_21a	5	12.4	4	13.0	7	3.5	4	5.6	3	4.1	1
Proc_21b	4	4.9	3	5.8	10	5.4	4	3.6	3	3.8	1
Proc_31a	3	2.4	2	1.6	4	2.0	4	2.4	3	1.9	1
Proc_31b	3	3.3	3	4.8	40	21.9	3	3.4	3	3.8	1
Proc_36a	3	5.8	3	9.1	139	108.9	3	5.9	3	9.1	1
Proc_36b	3	5.0	3	7.5	500	-	5	4.1	4	4.7	1

[a] It is the scalar values for PSC-QCUT and PSC-MQCUT.

We know that the general PSC cut is not tighter than the general OA cut. The scaled quadratic PSC cut is also looser than the scaled quadratic OA cut. When α of OA-QCUT is equal to 1 for the problems in Table 6, it should be also 1 for PSC-QCUT. For this problem set, OA-HCUT performs better on iterations than OA, but worse than OA-QCUT. In general, the number of iterations for OA-MHCUT is smaller than that of OA-HCUT, but longer solution times are required. The PSC with quadratic cuts

are generally better than PSC with linear cuts, especially for the csched2a and csched2 instances. For the test instances, PSC-MQCUT needs longer CPU time than PSC-QCUT when the number of iterations is similar. One issue noticed was that the PSC cuts for the Flay instances are very weak, which lead the three PSC methods to fail after surpassing the given maximum iteration limit.

6.4. Comparison to existing MINLP solvers

We also solve the 44 test problems with the BONMIN implementation of Outer-approximation 1.8 (B-OA) (Bonami et al., 2008), SBB (Bussieck and Vigerske, 2011) and SCIP 3.2 (Achterberg, 2009), which are either OA, Branch and Bound or Constraint Programming based MINLP solvers. This comparison gives an idea of how the proposed algorithms perform compared to available MINLP solvers. The detailed solution times for the solvers BONMIN, SBB and SCIP are given in the Appendix.

The CPU time performance profiles presented in Figure 8, Figure 9 and Figure 10 are based on the total of 44 test problems and the methodology proposed by Dolan and Moré (2002) for benchmarking optimization software. Each profile plots the cumulative amount of instances solved against the time at most a performance ratio factor τ to the fastest was given (Dolan and Moré, 2002).

The performance profiles for OA-QCUT, OA-MQCUT, OA, DICOPT, SBB, B-OA and SCIP are shown in Figure 8. We note that OA-QCUT and OA-MQCUT solve all the tested instances in the time limit, with OA-QCUT being more efficient for the section above 50% of the solved instances, while the rest being comparable. Both OA-QCUT and OA-MQCUT are better than OA according to this performance profile. SCIP continuously outperforms all methods until about 75% of the total problems. The performance profile shows that although the first solved problems are solved more efficiently by DICOPT, SBB and B-OA; the proposed algorithms could solve more instances given

the time limit.

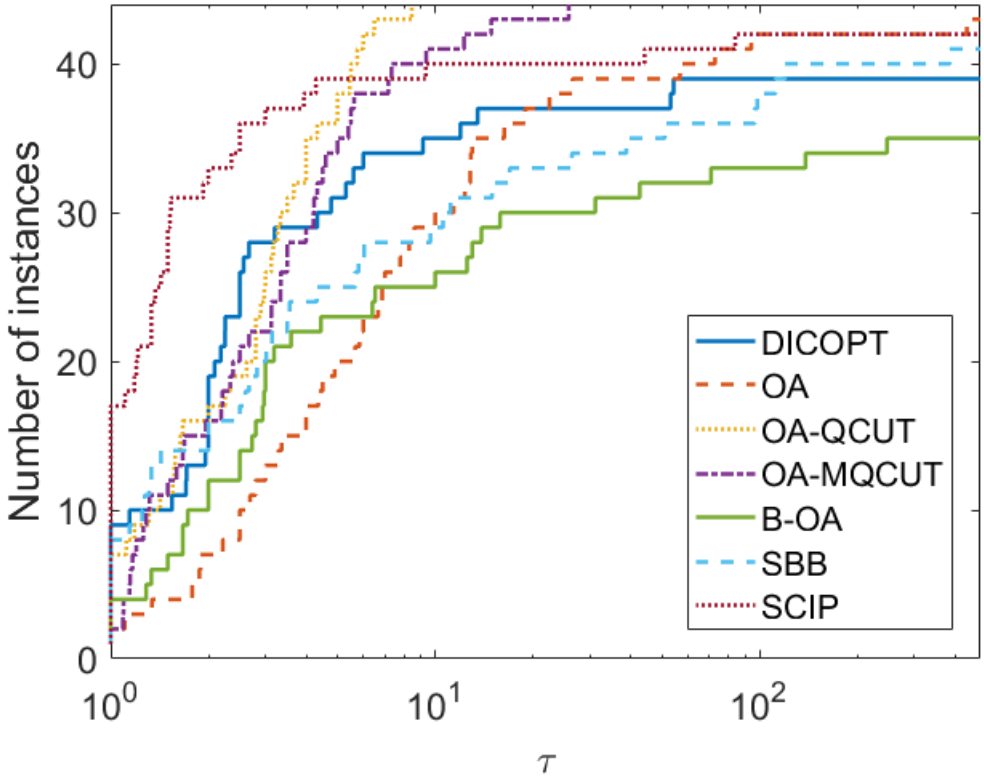


Figure 8. Performance profiles of OA-QCUT, OA-MQCUT, OA, DICOPT, SBB and B-OA

The CPU time performance profile for OA-HCUT, OA-MHCUT, OA, DICOPT, SBB, B-OA and SCIP are shown in Figure 9. OA-HCUT performs slightly better than OA-MHCUT, which both outperform the general OA and are the only algorithms able to solve all the tested instances in this comparison.

DICOPT, SBB and SCIP outperform OA-HCUT for the first 40% of the problems solved, and SCIP remains as the most efficient solver until about 75% of the instances. OA-HCUT performs better than all other compared methods for the last 25% of the instances.

Comparing the performances of the hybrid methods applied to OA (OA-HCUT and OA-MHCUT) with the quadratic OA (OA-QCUT and OA-MQCUT) we notice that the hybrid approach was slightly

less efficient that the quadratic only approach. This result can be supported by the fact that an important part of the tested instances has quadratic objective and constraints, which were exactly approximated by the quadratic cuts. The idea behind the hybrid approach is that for the initial iterations, a quadratic approximation is not required, and therefore it would be enough to use the linear approximation instead; which makes the master problem easier to solve.

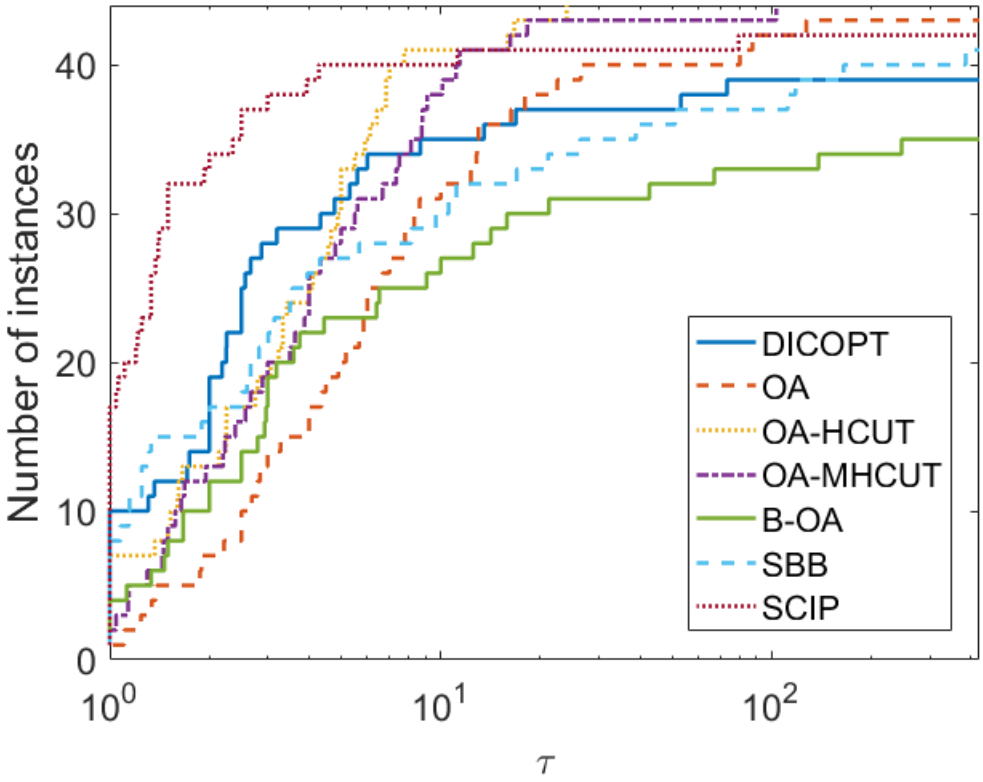


Figure 9. Performance profiles of OA-HCUT, OA-MHCUT, OA, DICOPT, SBB and BOA

Finally, the performance profiles regarding the CPU times for PSC-QCUT, PSC-MQCUT, PSC, DICOPT, SBB, B-OA and SCIP are shown in Figure 10, based on the 44 test instances.

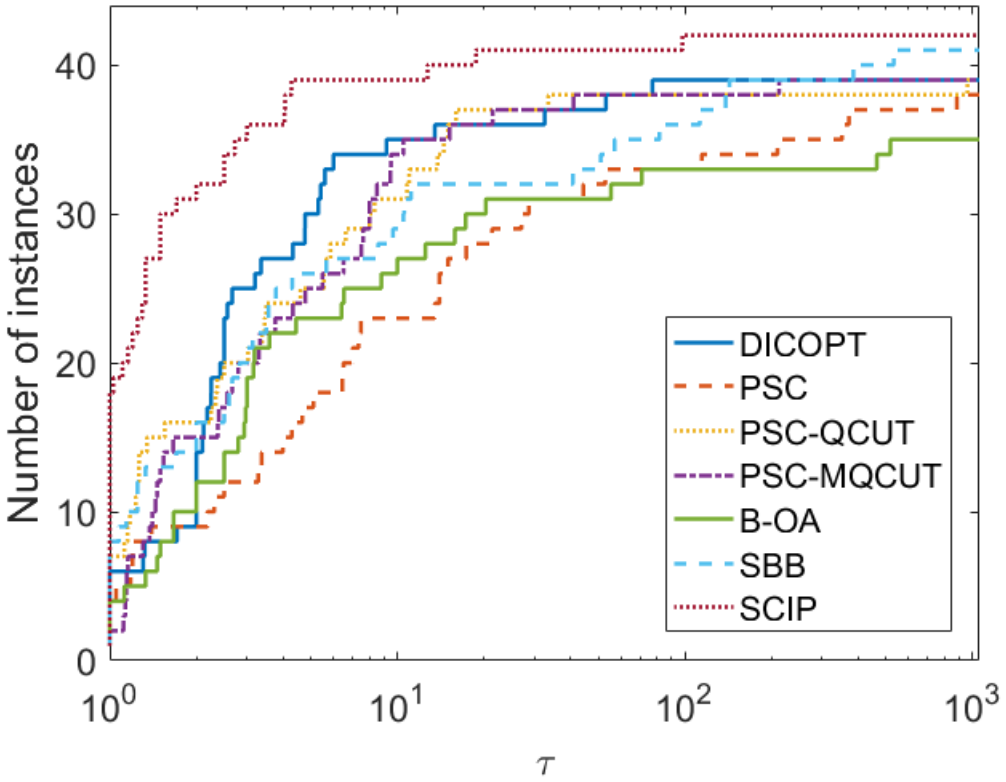


Figure 10. Performance profile of PSC-QCUT, PSC-MQCUT, PSC, DICOPT, SBB and B-OA

The performances of PSC-QCUT and PSC-MQCUT are similar; occasionally PSC-MQCUT outperforms PSC-QCUT, which both are better than general PSC method. SCIP outperforms all PSCs methods for the testing problem set.

7. Conclusions

In this paper, we proposed the scaled quadratic cuts for MINLP decomposition methods, namely OA and PSC. Based on the theoretical analysis, we derived the conditions that ensure that the scaled quadratic cuts can strictly underestimate a convex function. We designed the scaled quadratic cuts based on these observations for OA and PSC methods, which can obtain the optimal solutions for the convex MINLP problems. Integrating the strategies of scaled quadratic cuts, multi-generation cuts and linear cuts, we developed six improved MINLP methods, which are OA-QCUT, OA-MQCUT,

OA-HCUT, OA-MHCUT, PSC-QCUT and PSC-MQCUT. Numerical experiments show that OA and PSC with scaled quadratic cuts and multi-generation cuts can solve all the tested MINLP benchmark problems, and require fewer iterations and shorter CPU solution times, especially for MINLP problems with quadratic and highly nonlinear constraints, where the linear approximation is very poor.

After comparing the performance of each one of the six proposed MINLP methods, we can conclude that integrating 2^{nd} order scaled Taylor underestimators to the nonlinear constraints can improve the performance of the two decomposition algorithms, OA and PSC. The OA method appeared to outperform the PSC method in the tested instances, showing that at least for these instances the tradeoff between smaller but more MILP master problems solved is not preferable. The multi-generation cut strategy was useful to improve the performance of each decomposition algorithm, but it was less efficient in the compared tested instances than adding a single quadratic constraint. Using the hybrid linear-quadratic OA method (OA-HCUT) was less efficient that using quadratic cuts from the beginning. The most efficient proposed MINLP method was the OA-QCUT, which compared to other implementations of the OA method such as DICOPT and BONMIN, shows that the quadratic cuts can improve the overall efficiency of the method. Future work will aim at the solution of the general nonconvex MINLP problems using OA and PSC with scaled quadratic cuts.

Acknowledgments

The authors acknowledge financial support from the Fund for Innovative Research Groups of the National Natural Science Foundation of China (Grant No. 71321001), the Fund for the National Natural Science Foundation of China (Grant No. 61374203), the Fund for National Natural Science

Foundation of China for Major International Joint Research Project (Grant No. 71520107004), and the Fund for 111 Project (Grant No. B16009). The authors also acknowledge the Center for Advanced Process Decision-making at Carnegie Mellon University for financial support.

References

Achterberg, T., 2009. SCIP: Solving constraint integer programs. Math. Program. Comput. 1, 1–41. doi:10.1007/s12532-008-0001-1

Adjiman, C.S., Dallwig, S., Floudas, C.A., Neumaier, A., 1998. A global optimization method, αBB, for general twice-differentiable constrained NLPs — I. Theoretical advances. Comput. Chem. Eng. 22, 1137–1158. doi:10.1016/S0098-1354(98)00027-1

Alizadeh, F., Goldfarb, D., 2003. Second-order cone programming. Math. Program. 95, 3–51. doi:10.1007/s10107-002-0339-5

Belotti, P., Kirches, C., Leyffer, S., Linderoth, J., Luedtke, J., Mahajan, A., 2013. Mixed-Integer Nonlinear Optimization. Acta Numerica. 22, 1-131.

Berthold, T., Heinz, S., Vigerske, S., 2012. Extending a CIP Framework to Solve MIQCPs. Springer, New York, NY, pp. 427–444. doi:10.1007/978-1-4614-1927-3_15

Biegler, L.T., 2010. Nonlinear Programming: concepts, algorithms and applications to chemical processes. SIAM 397. doi:10.1137/1.9780898719383

Bliek, C., Bonami, P., Lodi, A., 2014. Solving Mixed-Integer Quadratic Programming problems with IBM-CPLEX : a progress report. Proc. Twenty-Sixth RAMP Symp. 171–180.

Bonami, P., Biegler, L.T., Conn, A.R., Cornuéjols, G., Grossmann, I.E., Laird, C.D., Lee, J., Lodi, A., Margot, F., Sawaya, N., Wächter, A., 2008. An algorithmic framework for convex mixed integer

nonlinear programs. Discret. Optim. 5, 186–204. doi:10.1016/j.disopt.2006.10.011

Bonami, P., Kilinç, M., Linderoth, J., 2012. Algorithms and software for convex mixed integer nonlinear programs. Mix. Integer Nonlinear Program. 1–39. doi:10.1007/978-1-4614-1927-3

Buchheim, C., Trieu, L., 2013. Quadratic outer approximation for convex integer programming with box constraints, in: Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). pp. 224–235. doi:10.1007/978-3-642-38527-8_21

Bussieck, M.R., Meeraus, A., 2004. General Algebraic Modeling System (GAMS), in: Modeling Languages in Mathematical Optimization, J. Kallrath (Ed.), Applied Optimization. Springer US, pp. 137–157.

Bussieck, M.R., Vigerske, S., 2011. MINLP Solver Software, in: Wiley Encyclopedia of Operations Research and Management Science. doi:10.1002/9780470400531.eorms0527

Dolan, E.D., Moré, J.J., 2002. Benchmarking optimization software with performance profiles. Math. Program. Ser. B 91, 201–213. doi:10.1007/s101070100263

Drud, A., 1985. CONOPT: A GRG code for large sparse dynamic nonlinear optimization problems. Math. Program. 31, 153–191. doi:10.1007/BF02591747

Duran, M.A., Grossmann, I.E., 1986. An outer-approximation algorithm for a class of mixed-integer nonlinear programs. Math. Program. 36, 307–339. doi:10.1007/BF02592064

Fletcher, R., Leyffer, S., 1994. Solving mixed integer nonlinear programs by outer approximation. Math. Program. 66, 327–349. doi:10.1007/BF01581153

Folland, G.B., 2002. Advanced calculus. Prentice Hall.

Geoffrion, A.M., 1972. Generalized Benders decomposition. J. Optim. Theory Appl. 10, 237–260.

doi:10.1007/BF00934810

Grossmann, I.E., 2002. Review of nonlinear mixed-integer and disjunctive programming techniques. Optim. Eng. 3, 227–252. doi:10.1023/A:1021039126272

Gupta, O.K., Ravindran, A., 1985. Branch and Bound Experiments in Convex Nonlinear Integer Programming. Manage. Sci. 31, 1533–1546. doi:10.1287/mnsc.31.12.1533

Horst, R., Tuy, H., 2013. Global Optimization: Deterministic Approaches, Springer. doi:10.1017/CBO9781107415324.004

IBM Corp., IBM, 2009. V12. 1: User's Manual for CPLEX. Int. Bus. Mach. Corp. 12, 481.

Kesavan, P., Allgor, R.J., Gatzke, E.P., Barton, P.I., 2004. Outer approximation algorithms for separable nonconvex mixed-integer nonlinear programs. Math. Program. 100, 517–535. doi:10.1007/s10107-004-0503-1

Leyffer, S., 2001. Integrating SQP and branch-and-bound for mixed integer nonlinear programming. Comput. Optim. Appl. 18, 295–309. doi:10.1023/A:1011241421041

Li, X., Tomasgard, A., Barton, P.I., 2011. Nonconvex Generalized Benders Decomposition for Stochastic Separable Mixed-Integer Nonlinear Programs. J. Optim. Theory Appl. 151, 425–454. doi:10.1007/s10957-011-9888-1

Méndez, C.A., Cerdá, J., Grossmann, I.E., Harjunkoski, I., Fahl, M., 2006. State-of-the-art review of optimization methods for short-term scheduling of batch processes. Comput. Chem. Eng. doi:10.1016/j.compchemeng.2006.02.008

Nesterov, Y.E., Todd, M.J., 1998. Primal-Dual Interior-Point Methods for Self-Scaled Cones. SIAM J. Optim. 8, 324–364. doi:10.1137/S1052623495290209

Quesada, I., Grossmann, I.E., 1992. An LP/NLP based branch and bound algorithm for convex

MINLP optimization problems. Comput. Chem. Eng. 16, 937–947. doi:10.1016/0098-1354(92)80028-8

Su, L., Tang, L., Grossmann, I.E., 2015. Computational strategies for improved MINLP algorithms. Comput. Chem. Eng. 75, 40–48. doi:10.1016/j.compchemeng.2015.01.015

Tang, L., Liu, J., Rong, A., Yang, Z., n.d. Theory and Methodology A mathematical programming model for scheduling steelmaking-continuous casting production.

Tang, L., Luh, P.B., Liu, J., Fang, L., 2002. Steel-making process scheduling using Lagrangian relaxation. Int. J. Prod. Res. 40, 55–70. doi:10.1080/00207540110073000

Tawarmalani, M., Sahinidis, N. V., 2004. Global optimization of mixed-integer nonlinear programs: A theoretical and computational study. Math. Program. 99, 563–591. doi:10.1007/s10107-003-0467-6

Trespalacios, F., Grossmann, I.E., 2014. Review of mixed-integer nonlinear and generalized disjunctive programming methods. Chemie-Ingenieur-Technik. doi:10.1002/cite.201400037

Viswanathan, J., Grossmann, I.E., 1990. A combined penalty function and outer-approximation method for MINLP optimization. Comput. Chem. Eng. 14, 769–782. doi:10.1016/0098-1354(90)87085-4

Wächter, A., Biegler, L.T., 2005. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Math. Program. 106, 25–57. doi:10.1007/s10107-004-0559-y

Appendix 1

As mentioned in the Section 4.2, in order to obtain a valid underestimator for a convex function using the scaled 2nd order Taylor expansion the scalar α has to be calculated using Eq. (14) or solving the following equivalent NLP problem.

$$\begin{aligned} \min_{x} \quad & \alpha = \frac{f(x) - f(x_0) - \nabla f(x_0)^T (x - x_0)}{(x - x_0)^T \nabla^2 f(x_0)(x - x_0)} \\ \text{s.t.} \quad & x \in F = \left\{ x \middle| a \le x \le b \right\} \setminus \{x_0\} \end{aligned} \tag{33}$$

In this work, instead of solving the previous NLP problem, we used Eq. (20) which considered the minimum of the function to be at the vertices.

In order to satisfy this condition we propose the following proposition.

Proposition. Let the scalar multiplying the quadratic term in the 2nd order Taylor expansion of the convex function $f(x)$ at the expansion point x_0 be defined as in Eq. (14) or as in Eq. (33). The value of the scalar can be calculated using Eq. (20) if the function $f(x)$ satisfies the following condition.

$$\begin{aligned} 2\nabla^2 f(x_0)(x - x_0)\left[f(x) - f(x_0) - \nabla f(x_0)^T (x - x_0) \right] \neq \\ \left[(x - x_0)^T \nabla^2 f(x_0)(x - x_0) \right] \left(\nabla f(x) - \nabla f(x_0) \right) \end{aligned} \tag{34}$$

For all $x \in X \subset \mathbb{R}^n$ in the region $F = \{x \mid a \le x \le b\} \setminus \{x_0\}$, where a and b are the lower and upper bounds of the variables x.

Proof.

The function defining α is the quotient of two convex functions, but is not convex itself. This fact is not neccesary to assure that its minimum is in one of the vertices of the hyper-box defined by the bounds of x. Since we are minimizing over the function α, we can assure that its minimum will be in one of the vertices if the function itself is component-wise monotonic. A component-wise

monotonic function is a function that is nonincreasing or nondecreasing in each variable independently.

The condition of the component-wise monotonicity can be ensured if none of the components of the gradient change their sign in the feasible set.

$$\begin{aligned} &\nabla\left(\frac{f(x)-f(x_0)-\nabla f(x_0)^T(x-x_0)}{(x-x_0)^T\nabla^2 f(x_0)(x-x_0)} \right) \quad \text{same sign} \\ &\text{s.t.} \quad x \in F = \left\{ x \middle| a \le x \le b \right\} \setminus \{x_0\} \end{aligned} \tag{35}$$

Calculating the gradient of the previous function we obtain the following expression.

$$\begin{aligned} &\frac{\left[(x-x_0)^T\nabla^2 f(x_0)(x-x_0)\right]\left(\nabla f(x)-\nabla f(x_0)\right)}{\left[(x-x_0)^T\nabla^2 f(x_0)(x-x_0)\right]^2} \\ &-\frac{2\nabla^2 f(x_0)(x-x_0)\left[f(x)-f(x_0)-\nabla f(x_0)^T(x-x_0)\right]}{\left[(x-x_0)^T\nabla^2 f(x_0)(x-x_0)\right]^2} \\ &\text{same sign} \\ &\text{s.t.} \quad x \in F = \left\{ x \middle| a \le x \le b \right\} \setminus \{x_0\} \end{aligned} \tag{36}$$

Since the function $f(x)$ is a convex function, the term $(x-x_0)^T\nabla^2 f(x_0)(x-x_0)$ is non-negative, therefore the denominator will be non-negative which avoids sign changes of all the components of the gradient. Taking into account that the function $f(x)$ is twice differentiable, then both the function and its gradient are continuous, and then the two terms can never be equal. This is the condition mentiones in Eq. (34), proving the given proposition. □

The condition given is stronger than requiring the function $f(x)$ to be convex, and it is strongly dependant of the expansion point x_0. Although the nonlinear functions involved in the test instances did satisfy this condition, e.g. quadratic constraints satisfy it trivially, it is not the general case.

Two counterexamples are given where the minimum of the α does not lie in one of the vertices, which results in an approximation which does not underestimate the function in the whole domain.

The first example has the following function in $\mathbb{R}$.

$$f(x) = b - \sqrt{(r^2 - (x-a)^2)} \tag{37}$$

where a, b and r are constants and the function $f(x)$ describes the semicircle or radius r and center (a,b). The given function is defined if the variable x has as bounds $x \in [a-r, a+r]$. Given a expansion point x_0 we obtain the following expression for calculating the α.

$$\frac{\alpha}{2} = \min_{x \in F = \{x | a \le x \le b\} \setminus \{x_0\}} \left\{ \frac{b - \sqrt{(r^2 - (x-a)^2)} - b + \sqrt{(r^2 - (x_0-a)^2)} - \frac{x-a}{\sqrt{(r^2 - (x_0-a)^2)}}(x-x_0)}{\frac{r^2}{(r^2 - (x_0-a)^2)^{\frac{3}{2}}}(x-x_0)^2} \right\} \tag{38}$$

Given the example of a semicircle of radius 3 and center in $(5,5)$, we obtain a plot of the function defining α in the Figure 11.

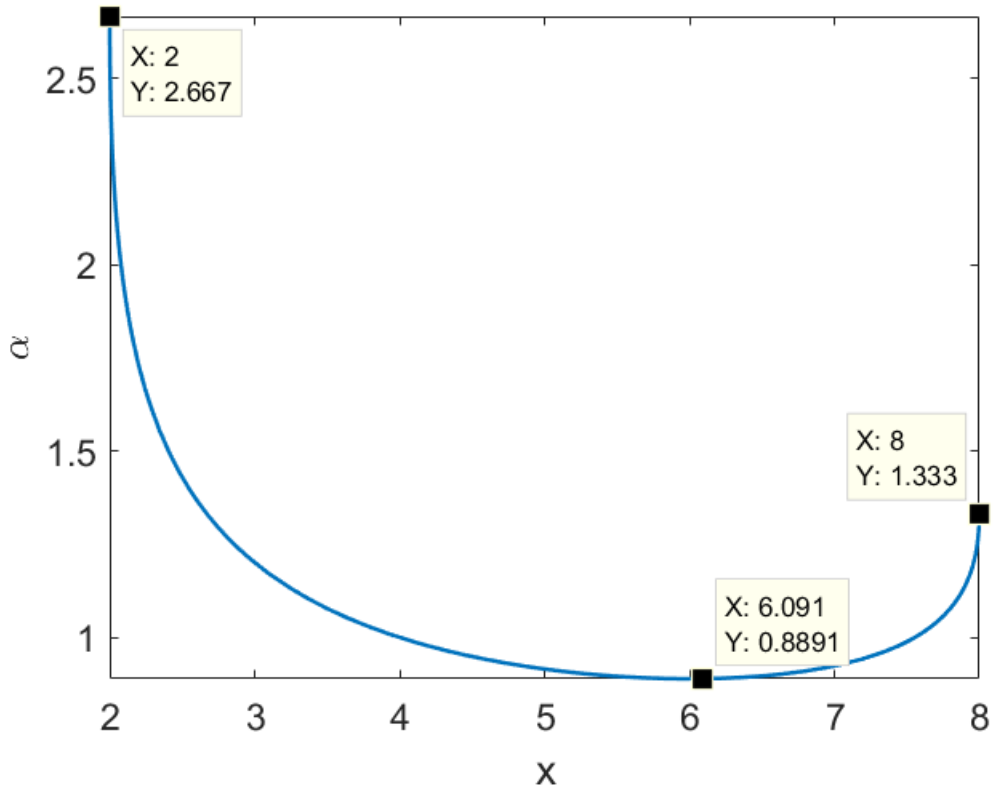


Figure 11. Scalar α for the function in Eq. (37) according to Eq. (38)

According to Eq. (20), the value of α should be 1.33 corresponding to the value at the vertex

$a+r=8$. In case we set $\alpha=1$, we would obtain the 2nd order Taylor approximation, but the minimum of the given function is not in the vertex nor is when $\alpha=1$.

The minimum of the given function can be analytically found and it appears at the point $x^{*}=2a-x_0$. The Figure 12 shows the original function with all its Taylor approximations.

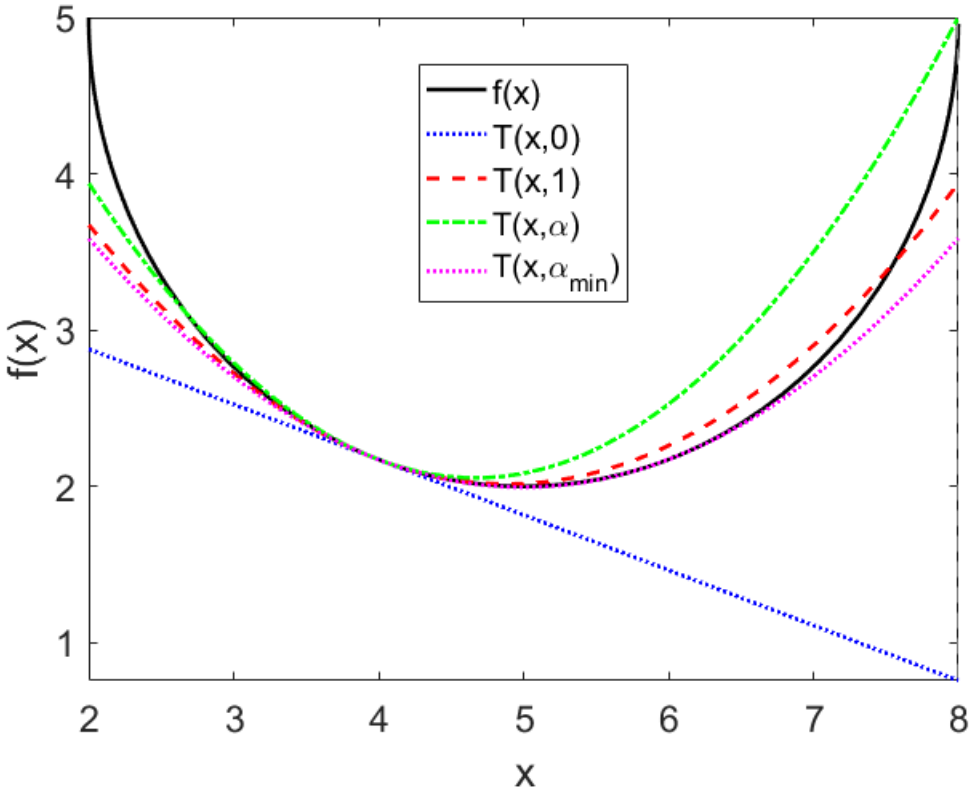


Figure 12. $f(x)=5-\sqrt{(3^2-(x-5)^2)}$ and its 1st order, 2nd order, scaled 2nd order with the scalar α calculated from the minimum at the vertices and the scaled 2nd order with the minimum scalar $\alpha_{\min}$

Note that the only Taylor approximations that are valid underestimators are the 1st order and the scaled 2nd order using the minimum α, as proposed in Proposition 2. This can be confirmed in Figure 12, where the difference between the function and its approximation is shown. In case the approximation is an underestimator, the difference is positive for the entire feasible region.

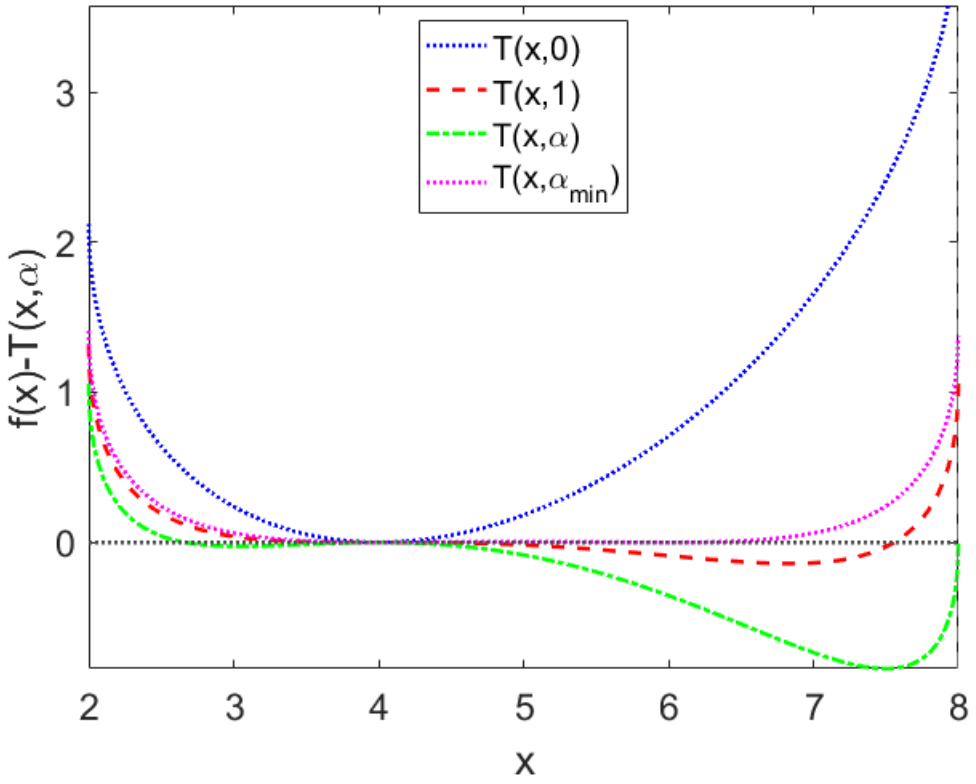


Figure 13. Difference between the function $f(x)=5-\sqrt{(3^2-(x-5)^2)}$ and the different order Taylor approximations

The second example is the following convex function in $\mathbb{R}^2$.

$$f(x_1,x_2)=x_1^{\,4}+x_2^{\,4} \tag{39}$$

And the following bounds on the variables $x_1\in[0.3,5], x_2\in[0.5,7]$, we expand the function at the point $x_1^{\,0}=4.9, x_2^{\,0}=4.9$.

The first order Taylor approximation of the function $f(x)$ is as follows.

$$T_1=-\frac{17294403}{5000}+\frac{117649}{250}(x_1+x_2) \tag{40}$$

The second order Taylor approximation is as follows.

$$\begin{aligned}T(x,\alpha)=&-\frac{17294403}{5000}+\frac{117649}{250}(x_1+x_2)+\\&\alpha\left[\left(x_1\frac{7203}{50}-\frac{352947}{500}\right)\left(x_1-\frac{49}{10}\right)+\left(x_2\frac{7203}{50}-\frac{352947}{500}\right)\left(x_2-\frac{49}{10}\right)\right]\end{aligned} \tag{41}$$

The function defining the scalar α is as follows.

$$\frac{\alpha}{2} = \min_{x \in F = \{x | a \le x \le b\} \setminus \{x_0\}} \left\{ \frac{x_1^4 + x_2^4 - \frac{17294403}{5000} - \frac{117649}{250}(x_1 + x_2)}{\left(x_1 \frac{7203}{50} - \frac{352947}{500}\right)\left(x_1 - \frac{49}{10}\right) + \left(x_2 \frac{7203}{50} - \frac{352947}{500}\right)\left(x_2 - \frac{49}{10}\right)} \right\} \quad (42)$$

According to the methodology proposed, the α should be calculated as in Eq. (20). Evaluating the function of α at the four vertices, we obtain the values reported in Table 7.

Table 7. Values of the scalar in the vertices of the counterexample.

Var.	**Vertex 1**	**Vertex 2**	**Vertex 3**	**Vertex 4**
x_1	0.3	0.3	5	5
x_2	0.5	7	0.5	7
α	0.5281	0.6582	0.536	0.316

Eq. (20) would calculate the value of the scalar α as 0.5281, corresponding to the vertex $(x_1^v, x_2^v) = (0.3, 0.5)$.

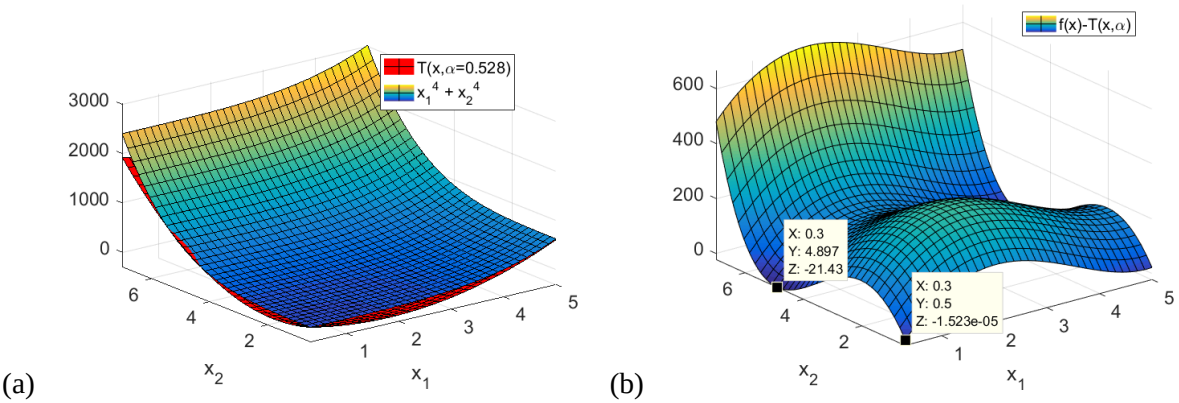


Figure 14. $f(x) = x_1^4 + x_1^4$ and scaled 2nd order Taylor approximation with $\alpha = 0.528$

(a) function values and (b) difference

As seen in Figure 14, the scaled second order approximation is not a valid underestimator for the function in Eq. (39). Note that at the point $(x_1, x_2) = (0.3, 4.9)$ .the difference from these two functions is -21.43, violating the condition in Eq. (15). This happens since the minimum of the

function defining α does not lie in a vertex, as seen in the Figure 15.

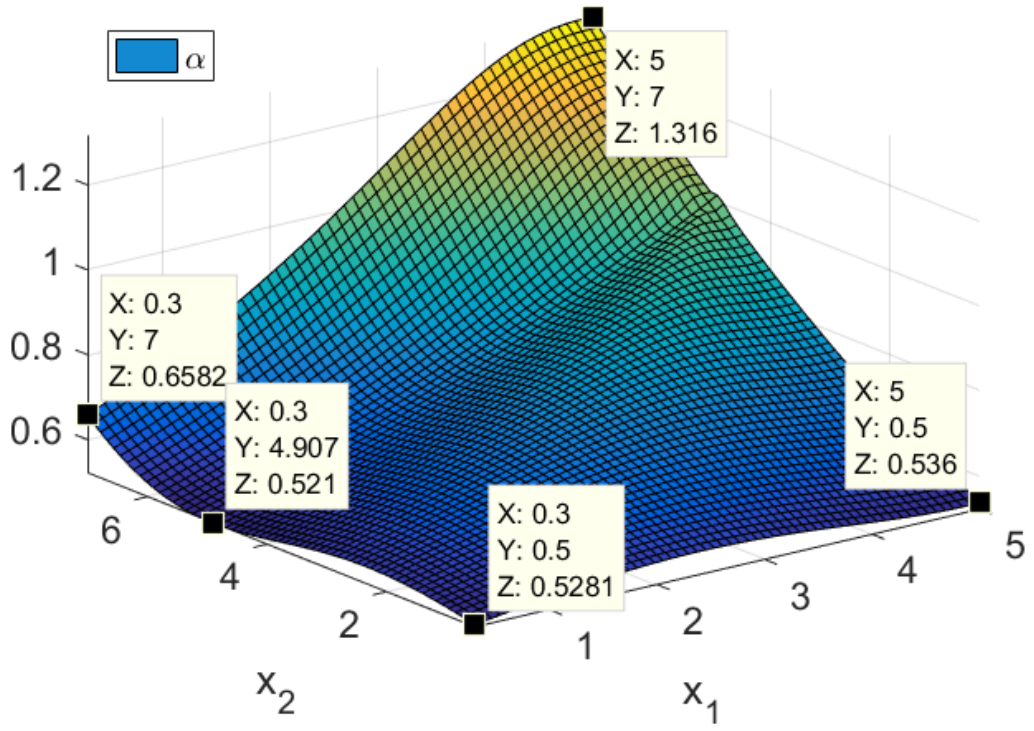
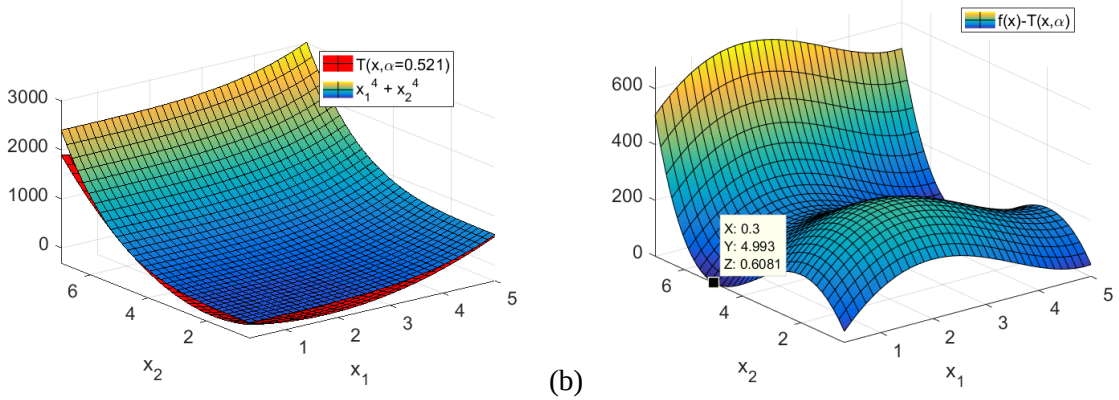


Figure 15. Scalar α for the function in Eq. (39) according to Eq. (42)

The function in Figure 15 is not coordinate-wise monotonic, so its minimum lies in the point $(x_1, x_2) = (0.3, 4.9)$, where $\alpha = 0.521$.

If we calculate the scaled 2nd order Taylor approximation using the minimum value of α we obtain a valid underestimator of the convex function $f(x)$ as Proposition 2 indicates shown in Figure 16.



(a) (b)

Figure 16. $f(x) = x_1^4 + x_1^4$ and scaled 2nd order Taylor approximation with $\alpha = 0.528$ (a) function values and (b) difference

Appendix 2

The problem statistics for the test problems of the Sections 6.1, 6.2, and 6.3 are listed in Table 8, Table 9 and Table 10 respectively.. Note that # means the number of something. Eqns are the abbreviation of the equations. Vars are the one of the variables. DVars represent the discrete variables. NZ mean non-zero coefficients. NNZ are the nonlinear matrix entries.

Table 8. Statistics of the 7 test problems in Section 6.1

Problems	**#Eqns**	**#Vars**	**#DVars**	**#NZ**	**#NNZ**
Synthes1	7	7	3	23	6
Synthes2	15	12	5	49	8
Synthes3	24	18	8	91	12
Gkocis	9	12	3	28	2
Alan	8	9	4	24	3
Ex1223b	10	8	4	32	17
St_e14	14	12	4	40	17

Table 9. Statistics of the 15 test problems in Section 6.2

Problems	**#Eqns**	**#Vars**	**#DVars**	**#NZ**	**#NNZ**
Clay42	93	53	32	319	64
Clay43	109	58	36	375	96
Clay44	125	62	40	431	128
Clay45	141	66	44	487	160
Clay52	138	82	50	483	80
Clay53	158	88	55	553	120
Clay54	178	92	60	623	160
Clay55	198	97	65	693	200
Slay04	55	45	24	189	8
Slay05	91	71	40	311	10
Slay06	136	103	60	463	12

Slay07	190	141	84	645	14
Slay08	253	185	112	857	16
Slay09	325	235	144	1099	18
Slay10	406	291	180	1371	20

Table 10. Statistics of the 22 test problems Section 6.3

Problems	**#Eqns**	**#Vars**	**#DVars**	**#NZ**	**#NNZ**
Batch03	20	20	9	53	10
Batch06	74	47	24	191	22
Batch08	218	102	60	547	40
Batch10	1020	279	129	2866	49
Batch12	1512	407	203	4256	59
Batch15	1782	446	203	5069	62
Batch20	2328	559	251	6664	67
Csched1a	23	29	15	78	7
Csched1	20	77	63	174	8
Csched2a	138	233	140	622	57
Csched2	138	401	308	958	58
Flay02	12	15	4	39	2
Flay03	25	27	12	87	3
Flay04	43	43	24	155	4
Flay05	66	63	40	243	5
Flay06	94	87	60	351	6
Proc_21a	62	48	21	205	15
Proc_21b	113	69	42	328	15
Proc_31a	102	77	41	371	31
Proc_31b	235	128	82	678	31
Proc_36a	122	92	46	431	36
Proc_36b	217	138	92	661	36

Appendix 3

In this Appendix we present the detailed solution times for the solvers BONMIN-OA, SBB and SCIP for the tested instances. The solution times are given in the Table 11, Table 12 and Table 13 according to the tested instances in the section 6.

Table 11. Solution CPU times(sec.) of BOMIN-BOA, SBB and SCIP solvers for the instances in Section 6.1

Problems	**BONMIN-BOA**	**SBB**	**SCIP**
Synthes1	0.4	0.3	0.4
Synthes2	0.5	0.2	0.3
Synthes3	0.6	0.3	0.4
Gkocis	0.2	0.2	0.2
Alan	0.2	0.2	0.3
Ex1223b	0.4	0.2	0.5
St_e14	0.5	0.2	0.3

Table 12. Solution CPU times(sec.) of BOMIN-BOA, SBB and SCIP solvers for the instances in Section 6.2

Problems	**BONMIN-BOA**	**SBB**	**SCIP**
Clay42	3.0	3.5	1.0
Clay43	4.0	5.1	0.9
Clay44	18.7	17.8	6.3
Clay45	93.3	44.9	14.3
Clay52	26.6	22.3	8.2
Clay53	20.6	14.5	1.3
Clay54	-	163.2	16.9
Clay55	-	389.7	36.9
Slay04	0.9	1.0	1.0
Slay05	1.9	1.4	1.3
Slay06	9.6	1.5	1.8
Slay07	46.7	8.7	2.3

Slay08	149.4	4.6	3.5
Slay09	-	118.2	7.9
Slay10	-	540.5	470.6

Table 13. Solution CPU times(sec.) of BOMIN-BOA, SBB and SCIP solvers for the instances in Section 6.3

Problems	**BONMIN-BOA**	**SBB**	**SCIP**
Batch03	0.2	0.5	0.5
Batch06	0.5	0.6	0.4
Batch08	1.4	1.3	1.0
Batch10	650.3	79.7	5.7
Batch12	1035.9	163.0	8.1
Batch15	-	171.4	15.3
Batch20	-	692.4	24.4
Csched1a	0.5	0.4	0.9
Csched1	0.7	0.8	3.0
Csched2a	127.4	3.6	-
Csched2	-	16.3	-
Flay02	0.6	0.5	0.4
Flay03	0.9	0.8	0.3
Flay04	23.8	8.2	1.9
Flay05	-	412.6	39.3
Flay06	-	-	1533.1
Proc_21a	4.7	5.7	1.6
Proc_21b	5.1	81.6	1.6
Proc_31a	6.0	263.4	2.2
Proc_31b	29.9	-	3.5
Proc_36a	38.0	1462.6	3.8
Proc_36b	70.8	-	7.0