J Glob Optim (2007) 39:79–100
DOI 10.1007/s10898-006-9121-1

ORIGINAL PAPER

A new global optimization method for univariate constrained twice-differentiable NLP problems

Min Ho Chang · Young Cheol Park · Tai-Yong Lee

Received: 28 April 2005 / Accepted: 2 November 2006 / Published online: 6 December 2006

Abstract In this paper, a new global optimization method is proposed for an optimization problem with twice-differentiable objective and constraint functions of a single variable. The method employs a difference of convex underestimator and a convex cut function, where the former is a continuous piecewise concave quadratic function, and the latter is a convex quadratic function. The main objectives of this research are to determine a quadratic concave underestimator that does not need an iterative local optimizer to determine the lower bounding value of the objective function and to determine a convex cut function that effectively detects infeasible regions for nonconvex constraints. The proposed method is proven to have a finite ϵ-convergence to locate the global optimum point. The numerical experiments indicate that the proposed method competes with another covering method, the index branch-and-bound algorithm, which uses the Lipschitz constant.

Keywords Global optimization · Difference of convex underestimator · Convex cut function · Univariate NLP

1 Introduction

The general univariate NLP problem is stated as follows:

$$\begin{aligned} &\min f(x) \\ &\text{subject to } g_j(x) \geq 0, \quad j = 1, \ldots, J, \\ &x \in X = [x_\mathrm{L}, x_\mathrm{U}], \end{aligned}$$

where $f(x)$ is the objective function, $g_j(x)$, $j = 1, \ldots, J$, are constraints, and x is the decision variable.

M. H. Chang · Y. C. Park · T.-Y. Lee (✉)
Department of Chemical and Biomolecular Engineering, Korea Advanced Institute of Science and Technology, 373-1 Guseong-dong, Yuseong-gu, Daejeon, 305-701, South Korea
e-mail: tylee@kaist.ac.kr

In order to solve the global optimization problem, many envelope methods have been proposed. The envelope is sometimes called an underestimator (or overestimator) for a minimization (or maximization) problem. Most envelope methods can be divided into two categories: convex underestimators and concave underestimators. If a convex underestimator is used, a local optimizer is needed in most cases. However, a concave underestimator is developed without a local optimizer.

Floudas et al. (2005) reviewed recent studies involving convexification techniques and convex envelopes for twice-differentiable NLPs. A commonly used convex underestimator is the one which is used in the αBB algorithm. The underestimator of αBB is generated to underestimate any twice-differentiable function, as proposed by Maranas and Floudas (1992). The αBB algorithm, which is a hybrid method involving a convex underestimator and a Branch-and-Bound (BB) algorithm was proposed and further developed by Maranas and Floudas (1994), Androulakis et al. (1995), Adjiman et al. (1996, 1998a, b), Adjiman and Floudas (1996), and Floudas (2000a). In the αBB algorithm, the nonconvex part of the objective function and constraints are divided into bilinear, trilinear, fractional, fractional trilinear, univariate concave, and general nonconvex terms, since, when constructing a convex underestimator for the function, the linear and convex terms do not require any transformation or an underestimator. There are improvements for various types of models and functions (Floudas et al. 2005), such as mixed-integer nonlinear models (Adjiman et al. 2000), differential-algebraic models (Esposito and Floudas 2000), grey-box and nonfactorable models (Meyer et al. 2002), explicit facets of convex and concave envelopes for trilinear functions (Meyer and Floudas 2004), and convex underestimator for trigonometric functions (Caratzoulas and Floudas 2005). For general nonconvex functions, Adjiman et al. (1998a) and Hertz et al. (1999) investigated the methods of calculating α using uniform and nonuniform diagonal shifts of the Hessian matrix, since a smaller α resulted in a tighter underestimator of the twice-differentiable function. A new class of improved convex underestimator, GαBB, has been proposed by Akrotirianakis and Floudas (2004a,b) and has shown improvements compared with the αBB. A spline αBB has been proposed by Meyer and Floudas (2005) with a piecewise quadratic function perturbation and a decrease in the difference between the objective function and the underestimator.

On the other hand, a concave underestimator, the Lipschitz underestimator, was introduced by Pijavskii (1972) and was developed by many researchers, such as Basso (1982), Mayne and Polak (1984), Hansen et al. (1992a,b), Hansen and Jaumard (1995), and Horst and Tuy (1996). The underestimator has a saw-tooth shape for the univariate objective function. Breiman and Culter (1993) proposed an envelope using a Lipschitzian derivative; MacLagan et al. (1996) proposed smoothed and accelerated BC (Breiman and Culter) envelopes; Sergeyev (1998) applied smooth auxiliary functions with a Lipschitzian derivative to make a smooth underestimator; and Gergel and Sergeyev (1999) improved the Lipschitzian derivative with sequential and parallel algorithms. Another concave underestimator is the polyhedral underestimator (Horst and Tuy 1996); it is utilized especially within concave minimization problems that have a concave objective function. The polyhedral underestimator is generated as a concave polyhedral using the concavity of the objective function. This underestimator is called a polyhedral annexation or an inner approximation.

For nonconvex constraints, several methods have been proposed. Some treat constraints implicitly, i.e., the constrained problem can be transformed into an unconstrained one using either a penalty, barrier, or augmented Lagrangian approach by

Elwakeil and Arora (1996) and Wang and Wah (1996). The Index Branch-and-Bound Algorithm (IBBA) for Lipschitz problems was proposed by Sergeyev et al. (2001, 2003) to handle the multiextremal constraints implicitly. This method converts the constrained problem into a discontinuous unconstrained problem using an index scheme without introducing additional parameters or variables. On the other hand, several methods have been developed from late 1990s to handle general constraints explicitly. Especially for the deterministic methods, there have been several methods that handle the general constraints differently. Using a reformulation of the original nonconvex constraints, relaxed problems become convex NLPs, as in Ryoo and Sahinidis (1995), Smith and Pantelides (1997), Zamora and Grossmann (1998), Adjiman et al. (1998b), and Floudas (2000b). Furthermore, several interval methods are proposed by Kearfott (1995), Ichida (1996), Byrne and Bogle (1999), and Jansson (2001) to handle general constrained nonconvex problems. These methods use the branch-and-bound algorithm and the information on constraints to reduce or discard infeasible subregions. Ryoo and Sahinidis (1995) proposed a feasibility-based range reduction technique, which introduced additional variables or bounded some variables based on the original problem constraints. Adjiman et al. (1998b) proposed a variable bound reduction technique to reduce selected variable bounds.

The αBB is based on the notion that the convex underestimator function is unimodal and has a unique (and global) minimum. The modified αBB framework for nonconvex functions, such as GαBB and spline αBB, are focused on making an improved underestimator compared with αBB (Floudas et al. 2005). However, if the global minimum of a certain underestimator can always be readily found, then it does not matter whether the underestimator is convex or not. In this study, a difference of convex underestimator (DCU), which is a continuous piecewise concave quadrature, is proposed and is focused on making an underestimator that does not need a local optimizer to determine the lower bound. The global minimum of the DCU is found easily via simple linear algebra. The lower bounding function can be updated using only one calculation of the objective function and its first derivative. Additionally, the upper bound is obtained while updating the lower bound. To handle nonconvex constraints, a convex cut function (CCF) is proposed. The CCF is generated using a similar methodology to the generation of the DCU. The infeasible region of the multiextremal constraints can be detected by generating the CCF.

In Section 2 and 3, the convex underestimator of αBB and DCU are, respectively, introduced for unconstrained problems. The methodology for generating the CCF for multiextremal constraints is introduced in Sect. 4. The overall algorithmic procedure is shown in Sect. 5. The results of the test problems are stated in Sect. 6, and our conclusions are presented in Sect. 7.

2 Convex underestimator of αBB

In general, an unconstrained single variable minimization problem can be defined by

$$\begin{gathered}\min f(x)\\ \text{subject to } x \in X = [x_{\mathrm{L}}, x_{\mathrm{U}}].\end{gathered}$$

The convex underestimator of the αBB algorithm with a general unconstrained NLP problem is defined by

$$\begin{aligned} L(x) &= f(x) + Q(x), \\ Q(x) &= \alpha(x_L - x)(x_U - x), \end{aligned} \tag{1}$$

where $f(x)$ is a nonlinear objective function and belongs to C^2, $L(x)$ is a relaxed convex underestimator, $Q(x)$ is a quadratic function convexifying $L(x)$, and α is a nonnegative constant.

Because $Q(x)$ in Eq. 1 is nonpositive over the entire interval $[x_L, x_U]$, $L(x)$ is a guaranteed underestimator of $f(x)$. Furthermore, $L(x)$ becomes a convex function since $Q(x)$ has a sufficiently large value of α:

$$L''(x) = f''(x) + 2\alpha.$$

Adjiman et al. (1998a) suggested several methods to calculate α. Most methods needed an interval second derivative $[f''(x)]$ such that $f''(x) \in [f''(x)]$ for any $x \in [x_L, x_U]$. The interval second derivative was calculated using the interval arithmetic of $f''(x)$:

$$\alpha = \max \left\{ 0, -\frac{1}{2} \min_{x \in X} [f''(x)] \right\}. \tag{2}$$

3 DCU for an unconstrained optimization problem

Let us consider $C(x)$, an intermediate function which is an underestimator of $L(x)$, and define a new underestimator of $f(x)$, $D(x)$:

$$D(x) = C(x) - Q(x). \tag{3}$$

When $C(x)$ is less than or equal to $L(x)$, $D(x)$ is less than or equal to $f(x)$, since $L(x)$ is a convex function defined using equations (1) and (2).

In Eq. 3, an intermediate function, $C(x)$, can be selected using a quadratic function:

$$C(x) = c_0 + c_1 x + c_2 x^2.$$

Then, Eq. 3 becomes

$$\begin{aligned} D(x) &= C(x) - Q(x) \\ &= (c_0 - \alpha x_L x_U) + (c_1 + \alpha x_L + \alpha x_U)\, x + (c_2 - \alpha)\, x^2. \end{aligned}$$

As shown in Table 1, $C(x)$ and $D(x)$ can be classified into five cases, depending on the value of c_2. Case I is the same as the case of Kim and Lee (2001)'s convex quadratic underestimator that was used as an accelerator for the underestimator in the αBB algorithm. Case II is similar to the bounding function of Lipschitz optimization (Hansen and Jaumard 1995). The present study is focused on case IV, since the underestimator of case IV can be obtained easily.

Table 1 Classification of underestimator

Case	c_2	$C(x)$	$D(x)$
I	$\alpha < c_2$	Convex quadratic	Convex quadratic
II	$c_2 = \alpha$	Convex quadratic	Linear
III	$0 < c_2 < \alpha$	Convex quadratic	Concave quadratic
IV	$c_2 = 0$	Linear	Concave quadratic
V	$c_2 < 0$	Concave quadratic	Concave quadratic

Let us consider case IV, where $c_2 = 0$. In this case, $C(x)$ and $D(x)$ are denoted as $R(x)$ and $S(x)$, respectively. The intermediate underestimator, $R(x)$, is a linear function and can be obtained by generating the tangential line of $L(x)$, since $L(x)$ is a convex function. The ultimate underestimator, $S(x)$, for the objective function is a quadratic concave function. Using $R(x)$, $S(x)$ can be generated using the following equation at the trial point, $x = \tilde{x}$:

$$R(x) = L'(\tilde{x})\left(x - \tilde{x}\right) + L(\tilde{x}).$$

$$S(x) = R(x) - Q(x). \tag{4}$$

When $R(x)$ is the tangential line of $L(x)$ at $x = \tilde{x}$, $S(x)$ is less than or equal to $f(x)$.

Now, let us construct $R(x)$ as an open polyhedron using a finite number of tangential lines of $L(x)$ to make a tight underestimator of $L(x)$:

$$R_k(x) = \max\left\{r(x;\tilde{x}_1), r(x;\tilde{x}_2), \ldots, r(x;\tilde{x}_k)\right\}, \tag{5}$$

where

$$r(x;\tilde{x}_i) = L'(\tilde{x}_i)\left(x - \tilde{x}_i\right) + L(\tilde{x}_i) \quad \text{for } i = 1, \ldots, k. \tag{6}$$

Note that $r(x;\tilde{x}_i)$ is the tangential line of $L(x)$ at $x = \tilde{x}_i$ and k is the number of tangential lines. Thus, $R_k(x)$ is a continuous piecewise linear function; it is an open polyhedron on each edge that is a segment of the tangential line of $L(x)$. $R_k(x)$ is convex and is clearly an underestimator of $L(x)$, since each $r(x;\tilde{x}_i)$ is an underestimator of $L(x)$. Then, the corresponding $S_k(x)$ is also generated by Eq. 4:

$$\begin{aligned} S_k(x) &= R_k(x) - Q(x) \\ &= \max\left\{r(x;\tilde{x}_1) - Q(x), r(x;\tilde{x}_2) - Q(x), \ldots, r(x;\tilde{x}_k) - Q(x)\right\} \\ &= \max\left\{s(x;\tilde{x}_1), s(x;\tilde{x}_2), \ldots, s(x;\tilde{x}_k)\right\}, \end{aligned} \tag{7}$$

where

$$s(x;\tilde{x}_i) = -\alpha(x - \tilde{x}_i)^2 + f'(\tilde{x}_i)(x - \tilde{x}_i) + f(\tilde{x}_i) \tag{8}$$

and $S_k(x)$ is a continuous piecewise concave quadratic function and is also an underestimator of $f(x)$. Let us define a set of the trial points, Z:

$$Z = \{\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_k\}.$$

If $\tilde{x}_1 < \tilde{x}_2 < \cdots < \tilde{x}_k$, a set of the intersections between $s(x;\tilde{x}_i)$ and $s(x;\tilde{x}_{i+1})$ can be defined as Y_k:

$$Y_k = \{x | s(x;\tilde{x}_i) = s(x;\tilde{x}_{i+1}) \quad \text{for} \quad i = 1, \ldots, k-1\}.$$

The upper bound is determined by selecting the minimum value among $f(\tilde{x}_i)$ for $i = 1, \ldots, k$, which are evaluated automatically while generating $r(x;\tilde{x}_i)$ for $i = 1, \ldots, k$:

$$f_k^U = \min_{x \in Z} f(x). \tag{9}$$

The lower bound is determined by selecting the minimum of $S_k(x)$, which is located at one of the vertices of $S_k(x)$:

$$f_k^L = \min_{x \in X} S_k(x) = \min_{x \in Y_k} S_k(x) = S_k(\check{x}), \tag{10}$$

where

$$\breve{x} = \arg\min_{x \in Y_k} S_k(x). \tag{11}$$

3.1 Basic algorithm

Step 1 Initialization:

(1) Given $f(x)$, x_L, and x_U, let $X = [x_\mathrm{L}, x_\mathrm{U}]$.
(2) $\alpha > \max\left\{0, -\frac{1}{2}\min_{x\in X}[f''(x)]\right\}$.
(3) Let $\tilde{x}_1 = x_\mathrm{L}$ and $\tilde{x}_2 = x_\mathrm{U}$.
(4) Calculate $s(x;\tilde{x}_i)$ for $i = 1, 2$ using Eq. 8.
(5) $S_0(x) = \max\{s(x;\tilde{x}_1), s(x;\tilde{x}_2)\}$.
(6) $Y_0 = \{x | s(x;\tilde{x}_1) = s(x;\tilde{x}_2)\}$ and $Z = \{\tilde{x}_1, \tilde{x}_2\}$.

Step 2 Repetition: for $k = 1, 2, \ldots,$

(1) Calculate $s(x;\tilde{x}_i) \forall \tilde{x}_i \in Y_{k-1}$ for $i = 1, \ldots, n_{k-1}$, using Eq. 8, where n_{k-1} is the number of elements of the set Y_{k-1}.
(2) $S_k(x) = \max\{S_{k-1}(x), s(x;\tilde{x}_1), s(x;\tilde{x}_2), \ldots, s(x;\tilde{x}_{n_{k-1}})\}$.
(3) $Y_k = \{x | S_{k-1}(x) = s(x;\tilde{x}_i)$ for $i = 1, 2, \ldots, n_{k-1}\}$.
(4) $Z = Z \cup Y_{k-1}$.

3.1.1 Convergence

After kth iteration of the basic algorithm, $n(Y_k) = 2^k$ and $n(Z) = 2^k + 1$. Let $x_\mathrm{L} = \tilde{x}_1 < \tilde{x}_2 < \cdots < \tilde{x}_{2^k+1} = x_\mathrm{U}$, where $\tilde{x}_i \in Z$ for $i = 1, 2, \ldots, 2^k + 1$, and $X_{ki} = [\tilde{x}_i, \tilde{x}_{i+1}]$, where $X = \bigcup_i X_{ki}$ for $i = 1, 2, \ldots, 2^k$.

Theorem 1 *Let $f(x)$ be twice-differentiable, $\beta_\mathrm{L} = \frac{1}{2}\inf_x f''(x) > -\alpha$, $\beta_\mathrm{U} = \frac{1}{2}\sup_x f''(x)$ $\forall x \in X = [x_\mathrm{L}, x_\mathrm{U}]$, and $l_{X_{ki}} = \tilde{x}_{i+1} - \tilde{x}_i$, length of X_{ki} for $i = 1, 2, \ldots, 2^k$. In the basic algorithm,*

1. $\lim_{k\to\infty} l_{X_{ki}} = 0 \, \forall i \in [1, 2, \ldots, 2^k]$ and
2. $\lim_{k\to\infty} (f(x) - S_k(x)) = 0$.

Proof For any subinterval X_{ki}, let us find the intersection between $s(x;\tilde{x}_i)$ and $s(x;\tilde{x}_{i+1})$, $\breve{x}_i$:

$$\frac{\breve{x}_i - \tilde{x}_i}{\tilde{x}_{i+1} - \tilde{x}_i} = \frac{f'(\tilde{x}_{i+1}) + \alpha(\tilde{x}_{i+1} - \tilde{x}_i) - \frac{f(\tilde{x}_{i+1}) - f(\tilde{x}_i)}{\tilde{x}_{i+1} - \tilde{x}_i}}{f'(\tilde{x}_{i+1}) - f'(\tilde{x}_i) + 2\alpha(\tilde{x}_{i+1} - \tilde{x}_i)} = \frac{\frac{1}{2}f''(\xi_2) + \alpha}{f''(\xi_3) + 2\alpha},$$

where

$$\begin{aligned} f(\tilde{x}_i) &= f(\tilde{x}_{i+1}) + f'(\tilde{x}_{i+1})(\tilde{x}_i - \tilde{x}_{i+1}) + \tfrac{1}{2}f''(\xi_2)(\tilde{x}_i - \tilde{x}_{i+1})^2, \\ f'(\tilde{x}_i) &= f'(\tilde{x}_{i+1}) + f''(\xi_3)(\tilde{x}_i - \tilde{x}_{i+1}) \end{aligned}$$

and $\tilde{x}_i \le \xi_2, \xi_3 \le \tilde{x}_{i+1}$. Because $f''(x) \in [2\beta_\mathrm{L}, 2\beta_\mathrm{U}] \, \forall x \in X$,

$$\frac{\breve{x}_i - \tilde{x}_i}{\tilde{x}_{i+1} - \tilde{x}_i} \ge \gamma$$

or

$$\frac{\tilde{x}_{i+1} - \breve{x}_i}{\tilde{x}_{i+1} - \tilde{x}_i} \le 1 - \gamma < 1, \tag{12}$$

where $\gamma = \frac{\beta_L + \alpha}{2(\beta_U + \alpha)}$. Similarly,

$$\frac{\breve{x}_i - \tilde{x}_i}{\tilde{x}_{i+1} - \tilde{x}_i} \le 1 - \gamma < 1. \tag{13}$$

At the $k + 1$st iteration of the basic algorithm, X_{ki} is divided into:

$$\begin{aligned} X_{(k+1)(2i-1)} &= [\tilde{x}_i, \breve{x}_i], \\ X_{(k+1)(2i)} &= [\breve{x}_i, \tilde{x}_{i+1}]. \end{aligned}$$

Using Eqs. 12 and 13, the length of $X_{(k+1)(2i-1)}$ and $X_{(k+1)(2i)}$ is reduced compared with the length of X_{ki}, at the worst case, with a ratio, $1 - \gamma$, which is determined as a value smaller than unity. Therefore, the lengths of X_{ki}, $\forall i$, converge to zero as k increases.

Let us define the difference between the objective function and the DCU, $\Delta_k(x)$, as follows:

$$\begin{aligned} \Delta_k(x) &= f(x) - S_k(x) \\ &= \min\{\delta_1(x), \delta_2(x), \ldots, \delta_{2^k}(x)\}, \end{aligned}$$

where

$$\delta_i(x) = \begin{cases} \delta_{i1}(x) = f(x) - s(x; \tilde{x}_i) & \text{for } x \in [\tilde{x}_i, \breve{x}_i], \\ \delta_{i2}(x) = f(x) - s(x; \tilde{x}_{i+1}) & \text{for } x \in [\breve{x}_i, \tilde{x}_{i+1}]. \end{cases}$$

The interval of $\delta_{i1}(x)$ is as follows:

$$0 \le \delta_{i1}(x) \le (\beta_U + \alpha)(\breve{x}_i - \tilde{x}_i)^2 \quad \text{for} \quad x \in [\tilde{x}_i, \breve{x}_i],$$

because $\delta''_{i1}(x) \in [2(\beta_L + \alpha), 2(\beta_U + \alpha)]$, $\delta_{i1}(\tilde{x}_i) = 0$, and $\delta'_{i1}(\tilde{x}_i) = 0$. The interval of $\delta_{i2}(x)$ is as follows:

$$0 \le \delta_{i2}(x) \le (\beta_U + \alpha)(\breve{x}_i - \tilde{x}_{i+1})^2 \quad \text{for } x \in [\breve{x}_i, \tilde{x}_{i+1}],$$

because $\delta''_{i2}(x) \in [2(\beta_L + \alpha), 2(\beta_U + \alpha)]$, $\delta_{i2}(\tilde{x}_{i+1}) = 0$, and $\delta'_{i2}(\tilde{x}_{i+1}) = 0$.

$$0 \le \delta_i(x) < (\beta_U + \alpha)(\tilde{x}_{i+1} - \tilde{x}_i)^2 \tag{14}$$

The interval of $\Delta_k(x)$,

$$0 \le \Delta_k(x) < \max\{(\beta_U + \alpha)(\tilde{x}_{i+1} - \tilde{x}_i)^2 | \quad \text{for } i = 1, 2, \ldots, 2^k\}.$$

Therefore, $\Delta_k(x)$ converges to zero as k increases, because the length of interval, X_{ki} $\forall i \in Y_k$, converges to zero as k increases. □

3.1.2 Convergence near a global optimum

In the basic algorithm, the function evaluations are performed at every vertex for every iteration. That means the underestimator of the basic algorithm, $S_k(x)$, is close to the objective function in the entire region. Using the 'cut-off test' based on Eqs. 9 and 10, however, the search region can be reduced. Using the reduction procedure of the search region, the search region narrows to the region including a global optimum, and the objective function becomes convex in this region. In order to show the convergence of objective function near the global optimum, let $\beta_{\mathrm{L}} = \frac{1}{2}\inf_x f''(x) > 0$, $\beta_{\mathrm{U}} = \frac{1}{2}\sup_x f''(x)\ \forall x \in X = [x_{\mathrm{L}}, x_{\mathrm{U}}]$, α is given in Eq. 2, and $x^* \in X$, where x^* is a global minimum and $f'(x^*) = 0$.

Let us execute the basic algorithm for the objective function, after the kth iteration, $n(Y_k) = 2^k$ and $n(Z) = 2^k + 1$. Let $x_{\mathrm{L}} = \tilde{x}_1 < \tilde{x}_2 < \cdots < \tilde{x}_{2^k+1} = x_{\mathrm{U}}$, where $\tilde{x}_i \in Z$ for $i = 1, 2, \ldots, 2^k + 1$, $X_{ki} = [\tilde{x}_i, \tilde{x}_{i+1}]$, and $X = \bigcup_i X_{ki}$ for $i = 1, 2, \ldots, 2^k$. Using Eqs. 12 and 13, the length of the subinterval X_{ki} is bounded as follows:

$$\gamma^k \Delta x \le l_{X_{ki}} \le (1-\gamma)^k \Delta x, \tag{15}$$

where $\Delta x = x_{\mathrm{U}} - x_{\mathrm{L}}$. An upper bound, f_k^{U}, can be bounded using Eq. 9:

$$f_k^{\mathrm{U}} = \min_{x \in Z} f(x) = f(\tilde{x}_m).$$

A lower bound on a subinterval X_{ki}, $f^{\mathrm{L}}_{X_{ki}}$, can be determined as follows:

$$f^{\mathrm{L}}_{X_{ki}} = \min\{f(\tilde{x}_i), f(\tilde{x}_{i+1}), s(\breve{x}_i; \tilde{x}_i)\}.$$

If $f^{\mathrm{L}}_{X_{ki}} = f(\tilde{x}_i)$ for $\tilde{x}_i > \tilde{x}_m$, X_{ki} can be discarded (cut-off-test):

$$0 \le \breve{x}_i - \tilde{x}_i \le \frac{f'(\tilde{x}_i)}{\alpha}, \tag{16}$$

since $s(\breve{x}_i; \tilde{x}_i) - f(\tilde{x}_i) = -\alpha(\breve{x}_i - \tilde{x}_i)^2 + f'(\tilde{x}_i)(\breve{x}_i - \tilde{x}_i)$ and $f(\tilde{x}_i) < f(\tilde{x}_{i+1})$. Let us consider a subinterval $X_{k(m-2^{k-2})}$ for $k \ge 2$ and $m + 2^{k-2} \le 2^k$. The condition of discarding $X_{k(m+2^{k-2})}$ can be obtained using Eqs. 13, 15, and 16:

$$(1-\gamma)^{k+2} \le \frac{2\beta_{\mathrm{L}}}{\alpha}(\gamma^3 - \gamma^{k+1}), \tag{17}$$

where

$$\breve{x}_{m+2^{k-2}} - \tilde{x}_{m+2^{k-2}} \le (1-\gamma)^{k+1}\Delta x \quad \text{and}$$

$$f'(\tilde{x}_{m+2^{k-2}}) \ge 2\beta_{\mathrm{L}}(\tilde{x}_{m+2^{k-2}} - x^*) \ge 2\beta_{\mathrm{L}}(\tilde{x}_{m+2^{k-2}} - \tilde{x}_{m+1}) \ge 2\beta_{\mathrm{L}}\frac{\gamma^3 - \gamma^{k+1}}{1-\gamma}\Delta x,$$

since $2\beta_{\mathrm{L}} \le f''(x) \le 2\beta_{\mathrm{U}}\ \forall x \in X$, $f(x^*) = 0$, $x^* \in [\tilde{x}_{m-1}, \tilde{x}_{m+1}]$, and $\tilde{x}_{m+2^{k-2}} - \tilde{x}_{m+1} \ge (\gamma^3 + \gamma^4 + \cdots + \gamma^k)\Delta x$. Similarly, a subinterval $X_{k(m-2^{k-2}-1)}$ for $m - 2^{k-2} - 1 \ge 1$ can be discarded using the previous condition. Therefore, the search region can be reduced as follows:

$$X = \bigcup_{i=\max\{1, m-2^{k-2}\}}^{\min\{m+2^{k-2}, 2^k\}} X_{ki}. \tag{18}$$

The length of the new search region can be bounded using Eq. 15:

$$l_X \le (1 - 2^{k-1}\gamma^k)\Delta x.$$

The difference between upper and lower bounds, d, can be bounded using Eqs. 14 and 15 as follows:

$$d = f_k^{\mathrm{U}} - f_k^{\mathrm{L}} < (2\beta_{\mathrm{U}} + \alpha)(1-\gamma)^{2k}\Delta x^2,$$

where

$$f_k^{\mathrm{U}} \le f^* + \beta_{\mathrm{U}}(1-\gamma)^{2k}\Delta x^2 \quad \text{and}$$

$$f_k^{\mathrm{L}} > f^* - (\beta_{\mathrm{U}} + \alpha)(1-\gamma)^{2k}\Delta x^2,$$

since $2\beta_{\mathrm{L}} \le f''(x) \le 2\beta_{\mathrm{U}}\ \forall x \in X$ and $f(x^*) = 0$.

Let μ be the smallest integer k that satisfies Eq. 17. After reducing the search region, $n(Y_\mu) = 2^{\mu-1}$ and $n(Z) = 2^{\mu-1} + 1$. At the next iteration, $2^{\mu-1}$ function evaluations are required. Using this procedure, the length of the search region can be repeatedly bounded. After additional iterations of similar procedure, the total number of function evaluations, n, is $2^\mu + 1 + (k-\mu)2^{\mu-1}$ for $k \ge \mu$, and the length of the new search region can be reduced as follows:

$$l_X \le (1 - 2^{\mu-1}\gamma^\mu)^{k-\mu+1}\Delta x.$$

The upper bound of the difference between upper and lower bounds, d_n^{U}, for the total number of function evaluations, n, is as follows:

$$d < d_n^{\mathrm{U}} = (2\beta_{\mathrm{U}} + \alpha)(1-\gamma)^{2k}\Delta x^2.$$

Definition 1 Suppose $\{p_n\}_{n=0}^\infty$ is a sequence that converges to p, with $p_n \ne p$ for all n. If positive constant λ and η exist with $\lim_{n\to\infty} \frac{|p_{n+1}-p|}{|p_n-p|^\eta} = \lambda$, then $\{p_n\}_{n=0}^\infty$ converges to p of order η, with asymptotic error constant λ.

$$\frac{d^{\mathrm{U}}_{n+2^{\mu-1}}}{d_n^{\mathrm{U}}} = \frac{(2\beta_{\mathrm{U}} + \alpha)(1-\gamma)^{2(k+1)}\Delta x^2}{(2\beta_{\mathrm{U}} + \alpha)(1-\gamma)^{2k}\Delta x^2} = (1-\gamma)^2. \tag{19}$$

Let $\eta = 1$ in definition 1. Then, the limit of the left-hand side of Eq. 19 is

$$\lim_{n\to\infty} \frac{d^{\mathrm{U}}_{n+2^{\mu-1}}}{d_n^{\mathrm{U}}} = \lim_{n\to\infty} \frac{d^{\mathrm{U}}_{n+1}}{d_n^{\mathrm{U}}}\frac{d^{\mathrm{U}}_{n+2}}{d_{n+1}^{\mathrm{U}}}\cdots\frac{d^{\mathrm{U}}_{n+2^{\mu-1}}}{d^{\mathrm{U}}_{n+2^{\mu-1}-1}} = \lambda_r^{2^{\mu-1}}, \tag{20}$$

where λ_r is the asymptotic error constant for the basic algorithm near a global optimum (relaxed algorithm). By Definition 1, we can say that the method shows linear convergence ($\eta = 1$) and the asymptotic error constant is determined using Eqs. 19 and 20:

$$\lambda_r = (1-\gamma)^{2^{2-\mu}}.$$

3.2 Algorithm 1

In Algorithm 1, the number of the objective function evaluation and its first derivative per one iteration is one, which is selected using Eq. 11: Algorithm 1 has the effect of a 'cut-off test' compared with the basic algorithm, since the vertex with a larger value of $S_k(x)$ than the upper bound in Eq. 9 cannot be selected. Algorithm 1 converges faster than the relaxed Algorithm near a global optimum, since the relaxed

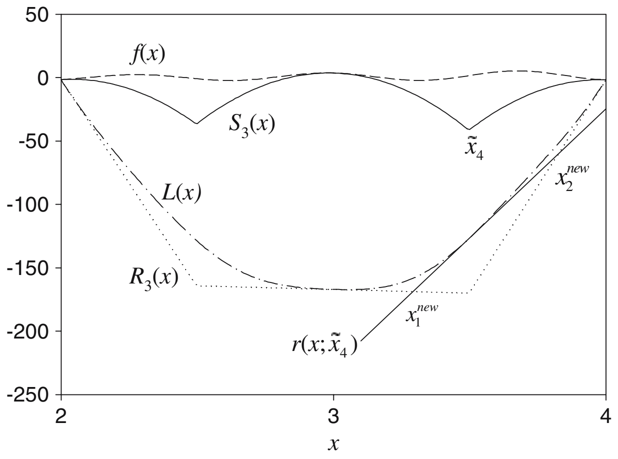

Fig. 1 Graphical representation of the proposed DCU

algorithm halves the number of subintervals after each $2^{\mu-1}$ function evaluations by removing the subintervals that do not include a global optimum. To find the solution of Eq. 11 easily, the elements of Y_k are sorted with the values of $S_k(x)$ decreasingly. $\check{x}$ is determined by the selection of the last element of Y_k.

If the stopping rule is not satisfied, let $k = k + 1$, select the next trial point $\tilde{x}_k = \check{x}$, and $Z = Z \cup \{\tilde{x}_k\}$. $r(x;\tilde{x}_k)$ is generated as a tangential line of $L(x)$ at $x = \tilde{x}_k$ using Eq. 6; $R_k(x)$ is determined using Eq. 5. Y_k is built by deleting one vertex from and adding two vertices to Y_{k-1}; the sequence of the elements in Y_k is updated with the values of $S_k(x)$, respectively:

$$Y_k = (Y_{k-1} - \{\tilde{x}_{k-1}\}) \cup \{x_1^{\text{new}}, x_2^{\text{new}}\} \cap X, \tag{21}$$

where $x_1^{\text{new}} < x_2^{\text{new}}$ and are the locations of the new vertices that are the intersecting points between $R_{k-1}(x)$ and $r(x;\tilde{x}_k)$. For $k = 1$,

$$\begin{aligned} x_1^{\text{new}} &= x_{\text{L}} \quad \text{and} \\ x_2^{\text{new}} &= x_{\text{U}}. \end{aligned} \tag{22}$$

For $k \geq 2$, x_1^{new} and x_2^{new} are determined by:

$$\begin{array}{lll} R_{k-1}(x_1^{\text{new}}) = r(x_1^{\text{new}};\tilde{x}_k), & x_2^{\text{new}} = +\infty & \text{for } \tilde{x}_k = x_{\text{L}} \\ R_{k-1}(x_1^{\text{new}}) = r(x_1^{\text{new}};\tilde{x}_k), & R_{k-1}(x_2^{\text{new}}) = r(x_2^{\text{new}};\tilde{x}_k) & \text{for } x_{\text{L}} < \tilde{x}_k < x_{\text{U}} \\ x_1^{\text{new}} = -\infty, & R_{k-1}(x_2^{\text{new}}) = r(x_2^{\text{new}};\tilde{x}_k) & \text{for } \tilde{x}_k = x_{\text{U}}. \end{array} \tag{23}$$

f_k^{U} can be obtained by selecting the minimum value between f_{k-1}^{U} and $f(\tilde{x}_k)$:

$$f_k^{\text{U}} = \min\left\{f_{k-1}^{\text{U}}, f(\tilde{x}_k)\right\}. \tag{24}$$

$S_k(x)$, f_k^{L}, and $\check{x}$ are determined using Eqs. 7, 10, and 11.

The proposed underestimator is shown in Fig. 1: at $k = 3$, $R_3(x)$ and $S_3(x)$ are obtained using Eqs. 5 and 7, respectively. The next trial point, $\check{x} = \tilde{x}_4$, is obtained using Eq. 11. $r(x;\tilde{x}_4)$ is calculated using Eq. 6 and the two new points, x_1^{new} and x_2^{new}, are generated using Eq. 23. The iterations continue until the stopping rule is satisfied.

Step 1 Initialization:

(1) Given $f(x)$, x_L, and x_U, let $X = [x_\mathrm{L}, x_\mathrm{U}]$.
(2) Determine α using Eq. 2.
(3) Let $\check{x} = (x_\mathrm{L} + x_\mathrm{U})/2$, $Y_0 = \{\check{x}\}$, and $Z = \emptyset$.
(4) Let $k = 1$ and $\tilde{x}_k = \check{x}$.

Step 2 Generation of DCU:

(1) Let $Z = Z \cup \{\tilde{x}_k\}$.
(2) Generate $r(x;\tilde{x}_k)$, $R_k(x)$, and $S_k(x)$ using Eqs. 5–7.
(3) Determine Y_k using Eqs. 21–23.
(4) Compute f_k^L using Eq. 10.
(5) If $k = 1$, then $f_k^\mathrm{U} = f(\tilde{x}_k)$, else compute f_k^U using Eq. 24.
(6) Determine the next trial point, $\check{x}$, using Eq. 11.
(7) Let $k = k + 1$.

Step 3 Convergence test:

(1) If $x > \check{x}\,\forall x \in Z$, $\check{x}_\mathrm{L} = x_\mathrm{L}$. Otherwise $\check{x}_\mathrm{L} = \max\{x | x < \check{x} \text{ for } x \in Z\}$.
(2) If $x < \check{x}\,\forall x \in Z$, $\check{x}_\mathrm{U} = x_\mathrm{U}$. Otherwise $\check{x}_\mathrm{U} = \min\{x | x > \check{x} \text{ for } x \in Z\}$.
(3) If $\check{x}_\mathrm{U} - \check{x}_\mathrm{L} < \epsilon$, terminate.
(4) Otherwise let $\tilde{x}_k = \check{x}$ and go to Step 2.

The stopping rule is selected with the same stopping rule of the sequential algorithm proposed by Gergel and Sergeyev (1999): a convergence accuracy, $\epsilon = 10^{-4}(x_\mathrm{U} - x_\mathrm{L})$.

4 CCF for a nonconvex constraint

If the trial point is located in an infeasible region while executing algorithm 1, the most violated nonconvex constraint, $g(x)$, which has the lowest function value among $g_j(x)$ for $j = 1, \ldots, J$ at the trial point, is selected and the CCF is generated to escape from the infeasible region. To generate the CCF, the overestimator of $g(x)$, $L_g(x)$, is generated by:

$$\begin{aligned} L_g(x) &= g(x) - Q_g(x) \geq g(x) \\ Q_g(x) &= \alpha_g (x_\mathrm{L} - x)(x_\mathrm{U} - x), \end{aligned} \tag{25}$$

where $g(x)$ is the most violated constraint, and α_g is chosen as a non-negative number large enough to make $L_g(x)$ concave. α_g can be determined by using a method similar to that for determining α in Sect. 2:

$$\alpha_g = \max\left\{0, \frac{1}{2}\max_{x \in X}[g''(x)]\right\}. \tag{26}$$

The CCF of $g(x)$, $S_g(x;\tilde{x})$, is generated using $R_g(x;\tilde{x})$, the tangential line of $L_g(x)$ when the trial point, $x = \tilde{x}$, is located in an infeasible region:

$$\begin{aligned} S_g(x;\tilde{x}) &= R_g(x;\tilde{x}) + Q_g(x) \\ &= \alpha_g (x - \tilde{x})^2 + g'(\tilde{x})(x - \tilde{x}) + g(\tilde{x}), \end{aligned} \tag{27}$$

where

$$R_g(x;\tilde{x}) = L_g'(\tilde{x})\left(x - \tilde{x}\right) + L_g(\tilde{x}). \tag{28}$$

When α_g is selected using Eq. 26 to make $L_g(x)$ concave in the given region and $R_g(x;\tilde{x})$ is a tangential line of $L_g(x)$, $S_g(x;\tilde{x})$ is greater than or equal to $g(x)$. A set defining the infeasible region, X^{inf}, can be determined by solving $S_g(x;\tilde{x}) < 0$:

$$X^{\text{inf}} = \{x | S_g(x;\tilde{x}) < 0 \text{ for } x \in X\}. \tag{29}$$

The search region, X, can be reduced as follows:

$$X = X - X^{\text{inf}}. \tag{30}$$

The underestimator of $f(x)$, $S_k(x)$, is a piecewise concave quadratic, but not continuous in the new search region. The set of locations of the vertices of $S_k(x)$, Y_k, should be updated to reflect the new search region:

$$Y_k = (Y_k - W_1) \cup W_2, \tag{31}$$

where $W_1 = \{x|S_g(x) < 0 \text{ for } x \in Y_k\}$ and $W_2 = \{x|S_g(x) = 0 \text{ for } x \in X\}$. The sequence of the elements of Y_k should be sorted by the values of $S_k(x)$ decreasingly. If $Y_k = \emptyset$, there is no search region: the problem is infeasible. Otherwise, the new trial point is obtained using Eq. 11 in the new search region. If $g(x)$ is concave, $\alpha_g = 0$ and $S_g(x) = R_g(x)$. $S_g(x)$ becomes a linear function and is the same as reducing a search region of the cutting plane method. The overall situation of generating the CCF is shown in Fig. 2. In Fig. 2a, $\tilde{x}$ is selected to be the next trial point. In Fig. 2b, $R_g(x;\tilde{x})$ and $S_g(x;\tilde{x})$ are generated using Eqs. 27 and 28. In Fig. 2c, $X^{\text{inf}} = [x_{\text{L}}^{\text{inf}}, x_{\text{U}}^{\text{inf}}]$ is determined using Eq. 29. Using Eq. 31, Y_k is updated by deleting $\tilde{x} \in W_1$ and adding $x_{\text{L}}^{\text{inf}}, x_{\text{U}}^{\text{inf}} \in W_2$.

5 Algorithmic procedure

The algorithm for an univariate NLP is constructed by adding a step involving a feasibility test to Algorithm 1. While executing Algorithm 1, the trial point is checked for feasibility. If the point is located in a feasible region, the DCU will be generated. Otherwise, the CCF will be generated to escape from the infeasible region.

5.1 Algorithm 2

Step 1 Initialization:

(1) Given $f(x)$, $g_j(x)$ for $j = 1, \ldots, J$, x_{L}, and x_{U}, let $X = [x_{\text{L}}, x_{\text{U}}]$.
(2) Determine α and α_{g_j} using Eqs. 2 and 26.
(3) Let $\check{x} = (x_{\text{L}} + x_{\text{U}})/2$, $Y_0 = \{\check{x}\}$, and $Z = \{\check{x}\}$.
(4) Let $k = 1$ and $\tilde{x}_k = \check{x}$.
(5) Go to Step 3.

Step 2 Feasibility test:

(1) Let $Z = Z \cup \{\tilde{x}_k\}$.
(2) If $g_j(\tilde{x}_k) + \epsilon_g \geq 0$ for $i = 1, \ldots, J$, go to Step 3.
(3) Otherwise, go to Step 4.

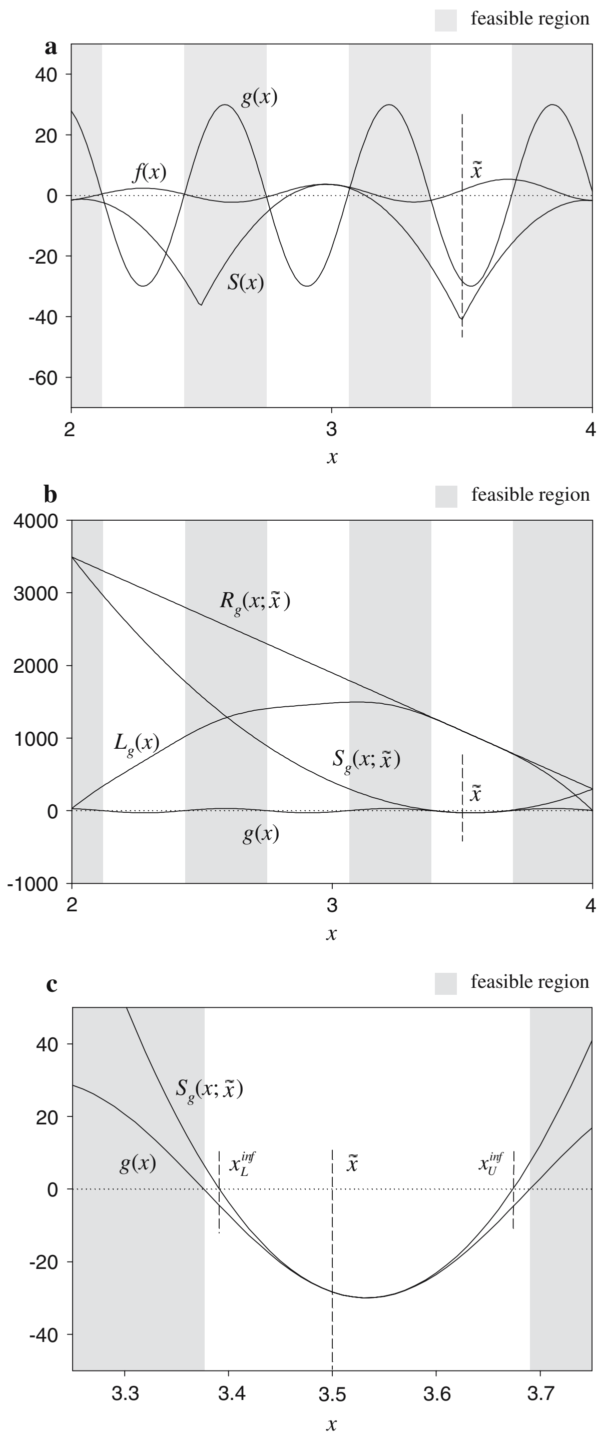

Fig. 2 Overall situation of generation of convex cut function. (**a**) Trial point, $\tilde{x}$, is infeasible (**b**) Generating convex cut function at $x = \tilde{x}$ (**c**) Magnified view of cutting region containing $\tilde{x}$

Step 3 Generation of DCU:

(1) Generate $r(x;\tilde{x}_k)$, $R_k(x)$, and $S_k(x)$ using Eqs. 5–7.
(2) Determine Y_k using Eqs. 21–23.
(3) Compute f_k^L using Eq. 10.
(4) If $k = 1$, then $f_k^{\mathrm{U}} = +\infty$, else f_k^{U} using Eq. 24.
(5) Determine the next trial point, $\breve{x}$, using Eq. 11.
(6) Let $k = k + 1$.
(7) Go to Step 5.

Step 4 Generation of CCF:

(1) Select the most violated constraint, $g(x)$.
(2) Generate $R_g(x;\tilde{x})$ and $S_g(x;\tilde{x})$ using Eqs. 27 and 28.
(3) Determine the infeasible region, X^{inf}, using Eq. 29.
(4) Reduce the search region, X, using Eq. 30.
(5) Determine Y_k using Eq. 31.
(6) Determine the next trial point, $\breve{x}$, using Eq. 11.
(7) Go to Step 5.

Step 5 Convergence test:

(1) If $x > \breve{x} \forall x \in Z, \breve{x}_{\mathrm{L}} = x_{\mathrm{L}}$. Otherwise $\breve{x}_{\mathrm{L}} = \max\{x | x < \breve{x} \text{ for } x \in Z\}$.
(2) If $x < \breve{x} \forall x \in Z, \breve{x}_{\mathrm{U}} = x_{\mathrm{U}}$. Otherwise $\breve{x}_{\mathrm{U}} = \min\{x | x > \breve{x} \text{ for } x \in Z\}$.
(3) If $\breve{x}_{\mathrm{U}} - \breve{x}_{\mathrm{L}} < \epsilon$, terminate.
(4) Otherwise let $\tilde{x}_k = \breve{x}$ and go to Step 2.

The difference between Step 2 of Algorithm 1 and Step 3 of Algorithm 2 is found in the search region. In Algorithm 2, the search region can be reduced using Eqs. 29 and 30. If the trial point is located in a feasible region, two new points, x_i^{new} for $i = 1, 2$, are generated in Eq. 23. When x_i^{new} is located in an infeasible region, x_i^{new} should not be included.

The stopping rule selected is that of the IBBA (Sergeyev et al. 2001) to compare Algorithm 2 with the IBBA: a convergence accuracy, $\epsilon = 10^{-4}(x_{\mathrm{U}} - x_{\mathrm{L}})$. A feasibility accuracy, ϵ_g, should be determined as a positive value: $\epsilon_g = 10^{-10}(x_{\mathrm{U}} - x_{\mathrm{L}})$, since the CCF is the tool for detecting the infeasible region.

5.2 Convergence

Using the DCU, Algorithm 1 converges to a global optimum for unconstrained optimization in Sect. 3. In this subsection, it will be shown how to detect the infeasible region using the CCF. To perform the detection of an infeasible region, an escape algorithm is proposed:

5.2.1 Escape algorithm

Step 1 Initialization:

(1) Given $g(x)$, x_{L}, and x_{U}, let $X = [x_{\mathrm{L}}, x_{\mathrm{U}}]$.
(2) α_g is calculated using Eq. 26.
(3) Choose any $\tilde{x} \in \{\tilde{x} | g(\tilde{x}) < 0 \text{ for } \tilde{x} \in X\}$.
(4) Generate $S_g(x;\tilde{x})$ and X^{inf} at $x = \tilde{x}$ using Eqs. 27 and 29.

(5) $x_{\mathrm{U0}}^{\mathrm{inf}} = \min\{x|x \in X^{\mathrm{inf}}\}$ and $x_{\mathrm{U0}}^{\mathrm{inf}} = \max\{x|x \in X^{\mathrm{inf}}\}$.

Step 2 Repetition: for $k = 1, 2, \ldots$

(1) $X_{\mathrm{L}}^{\mathrm{inf}} = \left\{x|S_g\left(x; x_{\mathrm{L}(k-1)}^{\mathrm{inf}}\right) < 0 \text{ for } x \in X\right\}$.
(2) $X_{\mathrm{U}}^{\mathrm{inf}} = \left\{x|S_g\left(x; x_{\mathrm{U}(k-1)}^{\mathrm{inf}}\right) < 0 \text{ for } x \in X\right\}$.
(3) $X^{\mathrm{inf}} = X^{\mathrm{inf}} \cup X_{\mathrm{L}}^{\mathrm{inf}} \cup X_{\mathrm{U}}^{\mathrm{inf}}$.
(4) $x_{\mathrm{L}k}^{\mathrm{inf}} = \min\{x|x \in X^{\mathrm{inf}}\}$ and $x_{\mathrm{U}k}^{\mathrm{inf}} = \max\{x|x \in X^{\mathrm{inf}}\}$.

$x_{\mathrm{U}k}^{\mathrm{inf}}$ of Step 2 is obtained using the solution of $S_g\left(x; x_{\mathrm{U}(k-1)}^{\mathrm{inf}}\right) = 0$:

$$\begin{aligned} x_{\mathrm{U}k}^{\mathrm{inf}} &= x_{\mathrm{U}(k-1)}^{\mathrm{inf}} + \frac{-g'\left(x_{\mathrm{U}(k-1)}^{\mathrm{inf}}\right) + \sqrt{g'\left(x_{\mathrm{U}(k-1)}^{\mathrm{inf}}\right)^2 - 4\alpha_g g\left(x_{\mathrm{U}(k-1)}^{\mathrm{inf}}\right)}}{2\alpha_g} \\ &= T\left(x_{\mathrm{U}(k-1)}^{\mathrm{inf}}\right). \end{aligned} \tag{32}$$

Theorem 2 *Given a constraint, $g(x) \geq 0$, and $X = [x_{\mathrm{L}}, x_{\mathrm{U}}]$, let $g(x)$ be twice-differentiable and α_g be obtained using* Eq. 26. *Consider $x_{\mathrm{UB}}^{\mathrm{inf}}$ and $x_{\mathrm{LB}}^{\mathrm{inf}}$ where $x_{\mathrm{LB}}^{\mathrm{inf}} < x_{\mathrm{UB}}^{\mathrm{inf}}$, $g(x_{\mathrm{LB}}^{\mathrm{inf}}) = g(x_{\mathrm{UB}}^{\mathrm{inf}}) = 0$, and $g(x) < 0 \forall x \in (x_{\mathrm{LB}}^{\mathrm{inf}}, x_{\mathrm{UB}}^{\mathrm{inf}})$. Given any $x_{\mathrm{U0}}^{\mathrm{inf}} \in (x_{\mathrm{LB}}^{\mathrm{inf}}, x_{\mathrm{UB}}^{\mathrm{inf}})$, if the sequence of $x_{\mathrm{U}k}^{\mathrm{inf}}$ is determined using* Eq. 32, *$x_{\mathrm{U}k}^{\mathrm{inf}}$ converges to $x_{\mathrm{UB}}^{\mathrm{inf}}$ as k increases.*

Proof $g'(x_{\mathrm{UB}}^{\mathrm{inf}}) \geq 0$ since $g(x_{\mathrm{UB}}^{\mathrm{inf}}) = 0$ and $g(x) < 0 \forall x \in (x_{\mathrm{LB}}^{\mathrm{inf}}, x_{\mathrm{UB}}^{\mathrm{inf}})$. Using Eq. 32, $x = x_{\mathrm{UB}}^{\mathrm{inf}}$ is the fixed-point of $T(x)$, since $g(x_{\mathrm{UB}}^{\mathrm{inf}}) = 0$ and $g'(x_{\mathrm{UB}}^{\mathrm{inf}}) \geq 0$. The sequence $x_{\mathrm{U}k}^{\mathrm{inf}}$ is strictly increasing and no interior point, $x \in (x_{\mathrm{LB}}^{\mathrm{inf}}, x_{\mathrm{UB}}^{\mathrm{inf}})$, can be a fixed-point of $T(x)$, since $g(x) < 0 \forall x \in (x_{\mathrm{LB}}^{\mathrm{inf}}, x_{\mathrm{UB}}^{\mathrm{inf}})$. The sequence $x_{\mathrm{U}k}^{\mathrm{inf}}$ is bounded as $x_{\mathrm{U}k}^{\mathrm{inf}} < x_{\mathrm{UB}}^{\mathrm{inf}}$, since $g(x) < S_g(x; x_{\mathrm{U}(k-1)}^{\mathrm{inf}}) \leq 0$ for $x \in (x_{\mathrm{U}(k-1)}^{\mathrm{inf}}, x_{\mathrm{U}k}^{\mathrm{inf}}]$. Therefore, the sequence $x_{\mathrm{U}k}^{\mathrm{inf}}$ converges to $x_{\mathrm{UB}}^{\mathrm{inf}}$.

Similarly, the sequence $x_{\mathrm{L}k}^{\mathrm{inf}}$ of Step 2 is strictly decreasing and converges to $x_{\mathrm{LB}}^{\mathrm{inf}}$. The escape algorithm can detect an infeasible region, $\left(x_{\mathrm{LB}}^{\mathrm{inf}}, x_{\mathrm{UB}}^{\mathrm{inf}}\right)$.

If $g'\left(x_{\mathrm{UB}}^{\mathrm{inf}}\right) > 0$, a Taylor series expansion of Eq. 32 in the vicinity of $x_{\mathrm{UB}}^{\mathrm{inf}}$ is given as follows:

$$x_{\mathrm{U}k}^{\mathrm{inf}} = x_{\mathrm{UB}}^{\mathrm{inf}} - \frac{2\alpha_g - g''(\xi_4)}{2g'(\xi_4)}\left(x_{\mathrm{U}(k-1)}^{\mathrm{inf}} - x_{\mathrm{UB}}^{\mathrm{inf}}\right)^2,$$

where $x_{\mathrm{U}(k-1)}^{\mathrm{inf}} \leq \xi_4 \leq x_{\mathrm{UB}}^{\mathrm{inf}}$. As the iteration progresses, ξ_4 converges to $x_{\mathrm{UB}}^{\mathrm{inf}}$, and Definition 1 states that the sequence $x_{\mathrm{U}k}^{\mathrm{inf}}$ converges quadratically. If $g'\left(x_{\mathrm{UB}}^{\mathrm{inf}}\right) = 0$, $x_{\mathrm{UB}}^{\mathrm{inf}}$ is a multiple root of $g(x)$ and $x_{\mathrm{U}k}^{\mathrm{inf}}$ converges linearly, but the proof is omitted here. The same is true for $x_{\mathrm{LB}}^{\mathrm{inf}}$.

In Algorithm 2, the escape algorithm is improved efficiently as the step of 'Generation of CCF' is performed with the candidate of the global optimum, $\check{x}$, using Eq. 11.

Table 2 The asymptotic error constants for unconstrained problems using the relaxed Algorithm (λ_r) and Algorithm 1 (λ_{DCU})

Problem	α	μ	a	λ_r	λ_{DCU}
UC01	1025	20	−0.5887	1.0000	0.2578
UC02	5.4793	4	−0.4477	0.8758	0.3567
UC03	174.1	4	−0.1404	0.9576	0.7237
UC04	0	2	−0.5167	0.8394	0.3043
UC05	334.38	4	−0.4394	0.8726	0.3635
UC06	2.0305	6	−0.3792	0.9725	0.4176
UC07	5.4926	4	−0.3551	0.8738	0.4415
UC08	117.1	4	−0.1645	0.9582	0.6848
UC09	0.69082	4	−0.3683	0.8751	0.4282
UC10	2.7673	4	−0.5358	0.8869	0.2912
UC11	3	6	−0.1690	0.9855	0.6777
UC12	1.8634	6	−0.1678	0.9862	0.6795
UC13	0	2	−0.6043	0.8830	0.2487
UC14	10.181	4	−0.4545	0.8935	0.3512
UC15	6.5431	18	−0.0880	1.0000	0.8166
UC16	0	2	−0.5676	0.9726	0.2706
UC17	108	49	−0.2721	1.0000	0.5345
UC18	1	3	−0.6021	0.7071	0.2500
UC19	4.5	4	−0.4240	0.8818	0.3767
UC20	0.13756	5	−0.4155	0.9411	0.3842

6 Results and discussion

The proposed method was numerically tested with two different aspects:

1. Algorithm 1 was performed for unconstrained optimization problems. Twenty test problems (UC01–UC20) were adopted from Hansen and Jaumard (1995). The numerical experiments were performed for the convergence near the global optimum and for comparison with the results of BC (Breiman and Culter 1993) and SA (Gergel and Sergeyev 1999). BC and SA construct nonsmooth underestimators, and the underestimator of Algorithm 1 is also a non smooth underestimator. MacLagan et al. (1996) and Sergeyev (1998) construct better smooth underestimators using smooth auxiliary functions. In this paper, however, a comparison with smooth underestimators is not considered, since the smooth underestimators are a different type of underestimators compared with BC, SA, and Algorithm 1.
2. Algorithm 2 was performed for nonconvex constrained optimization problems. The ten 'differentiable test problems', problems 1–10 (NC01–NC10), were adopted from Famularo et al. (2001), and numerical experiments were compared with IBBA (Sergeyev et al. 2001). The IBBA was developed to be able to apply nondifferentiable objective functions and constraints. IBBA can be applied to solve a wider range of global optimization problems compared with Algorithm 2, which uses the information of the first and second derivatives: the first derivative makes a tighter underestimator as iteration increases, and the second derivative should be determined priorly like the Lipschitz constants of IBBA. In this paper, however, a numerical comparison is performed to introduce a new type algorithm for global optimization problems.

For solving unconstrained optimization problems (UC01–UC20), the value of α should be determined priorly. The values of α, shown in Table 2, were determined

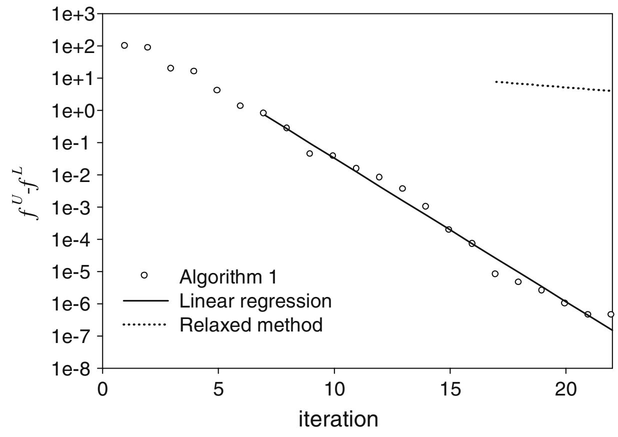

Fig. 3 Result of Algorithm 1 using DCU for problem UC02

using Eq. 2. In Sect. 3.1, the relaxed algorithm, the basic algorithm for the convergence near a global optimum, is applied to show that the method fits well to the linear convergence.

To calculate the asymptotic error constant of the relaxed Algorithm, λ_r, the integer index, μ, which is the smallest integer k that satisfies Eq. 17, can be obtained using the following search region which satisfies $\beta_\mathrm{L} > 0$:

$$X = \left[\frac{1}{2}\left(x^* + x_\mathrm{L}^c\right), \frac{1}{2}\left(x^* + x_\mathrm{U}^c\right)\right],$$

where

$$x_\mathrm{L}^c = \max\left\{x_\mathrm{L}, \max\{x|f''(x) = 0 \text{ for } x < x^*\}\right\} \quad \text{and}$$

$$x_\mathrm{U}^c = \min\left\{x_\mathrm{U}, \min\{x|f''(x) = 0 \text{ for } x > x^*\}\right\}.$$

To obtain the asymptotic error constants of Algorithm 1, linear regression was used for the log-scale value of the difference between the upper and lower bounds versus the number of function evaluations. $\epsilon = 10^{-4}(x_\mathrm{U} - x_\mathrm{L})$ was selected as the stopping condition.

Figure 3 shows the convergence characteristics of problem UC02, where the circles represent $f^\mathrm{U} - f^\mathrm{L}$ generated by Algorithm 1. It can be intuitively said that, as computation progresses beyond eight iterations, the circles align in a straight line on this log-linear plot. Linear regression yields the slope, $a = -0.4477$, and the asymptotic error constant, $\lambda_\mathrm{DCU} = 10^{-0.4477} = 0.3567$. The dotted line represents the convergence characteristics of the relaxed algorithm, which yields the asymptotic error constant, 0.8758 with $\mu = 4$. Problems, UC03, UC08, UC11, UC12, and UC17, have multiple global optima. The numbers of the global optima are three for problems UC03 and UC08, and two for problems UC11, UC12, and UC17. When a problem has multiple global optima, the asymptotic error constant calculated by the relaxed algorithm should be corrected by the number of optima before comparison. In Table 2, the results for the convergence of Algorithm 1 are shown and the asymptotic error constants of Algorithm 1, λ_DCU, are less than those of the relaxed algorithm, λ_r, for all

Table 3 Numerical results of Algorithm 1 (DCU)

Problem	x_{DCU}	f_{DCU}	DCU	x^* (Hansen and Jaumard 1995)	f^* (Hansen and Jaumard 1995)	BC (Gergel and Sergeyev 1999)	SA (Gergel and Sergeyev 1999)
UC01	10.000	29763	14	10.000	29763	26	27
UC02	5.1459	1.8996	23	5.1457	1.8996	22	27
UC03	−6.7744	12.031	99	−6.7746	12.031	104	98
UC04	2.8681	3.8505	16	2.8680	3.8505	25	27
UC05	0.9661	1.4891	29	0.9661	1.4891	33	23
UC06	0.6797	0.8242	37	0.6796	0.8242	38	39
UC07	5.1997	1.6013	25	5.1998	1.6013	25	25
UC08	5.4828	14.508	67	5.4829	14.508	86	88
UC09	17.039	1.9060	26	17.309	1.9060	25	26
UC10	7.9787	7.9167	18	7.9787	7.9167	25	25
UC11	2.0946	1.5000	44	2.0940	1.5000	45	41
UC12	4.7123	1.0000	42	4.7120	1.0000	43	37
UC13	0.7072	1.5874	14	0.7071	1.5874	278	89
UC14	0.2250	0.7887	23	0.2249	0.7887	30	30
UC15	2.4142	0.0355	81	2.4142	0.0355	81	47
UC16	1.5908	−7.5159	16	1.5907	−7.5159	87	75
UC17	3.0002	−7.0000	29	3.0000	−7.0000	70	65
UC18	2.0001	0.0000	16	2.0000	0.0000	20	21
UC19	5.8730	7.8157	19	5.8729	7.8157	19	21
UC20	1.1952	0.0635	30	1.1951	0.0635	20	32
Average			33.4			55.1	43.15

problems. Having a smaller value of asymptotic error constant, the convergent rate increases. In other words, Algorithm 1 has a faster convergent rate than the relaxed algorithm and shows a linear convergence.

The numerical results of Algorithm 1 are summarized in Table 3: x_{DCU} and f_{DCU} represent the global optimum using Algorithm 1 and 'DCU' represents the number of trials of Algorithm 1: one trial includes one evaluation of a function and its derivative. x^* and f^* are the global optimum and its objective function value cited by Hansen and Jaumard (1995), respectively. 'BC' and 'SA' represent the number of trials of BC and SA cited by Gergel and Sergeyev (1999). The computational load, in terms of the number of trials, decreases to about 61 and 77% compared with BC and SA, respectively.

For solving nonconvex optimization problems (NC01–NC10), The values of α and α_{g_j} for $j = 1,2,3$ should be determined priorly (see Table 4). These values are used to generate the DCU and CCF in Eqs. 1 and 25; they are also determined using Eqs. 2 and 26 for $f(x)$ and $g_j(x)$ for $j = 1,\ldots,J$. All constraints are evaluated and the most violated constraint that has the lowest function value is selected to generate the CCF.

Table 5 contains the information of the global optimum (Famularo et al. 2001) and the summary of the results of the IBBA (Sergeyev et al. 2001). x^* and f^* are the global optimum and its objective function value, respectively, and x_{IBBA} and f_{IBBA} are the corresponding values obtained by IBBA. $N_{\mathrm{total}}^{\mathrm{IBBA}}$ is the total number of function evaluations using IBBA.

Table 6 shows the results obtained using Algorithm 2 and the columns in the table have the following meaning:

Table 4 The values of α and α_{g_j} for $j = 1,2,3$ for $f(x)$ and $g_j(x)$ for $j = 1,2,3$

Problem	α	α_{g1}	α_{g2}	α_{g3}
NC01	21.125	12.34	–	–
NC02	3.275	1.65	–	–
NC03	20.8	2.4	–	–
NC04	34	124.4	45.75	–
NC05	1	0.069	5.25	–
NC06	12.8	61	2.5	–
NC07	21.682	16.35	25.38	–
NC08	39.75	22	900	60
NC09	7.55	0.5	1.5	0.5
NC10	12.4	15	2.375	7

Table 5 Global optimum and results of IBBA

Problem	Famularo et al. (2001)		IBBA (Sergeyev et al. 2001)		
	x^*	f^*	x_{IBBA}	f_{IBBA}	$N_{\text{total}}^{\text{IBBA}}$
NC01	1.0573	−7.6128	1.0573	−7.6123	36
NC02	1.0160	5.4606	1.0156	5.4616	248
NC03	−5.9921	−2.9460	−5.9918	−2.9427	84
NC04	2.4596	2.8408	2.4597	2.8408	1459
NC05	8.8573	−1.2730	9.2850	−1.2748	402
NC06	2.3240	−1.6851	2.3240	−1.6852	228
NC07	−0.7746	−0.3301	−0.7747	−0.3301	794
NC08	−1.1272	−6.6006	−1.1272	−6.6006	536
NC09	4.0000	1.9222	4.0005	1.9222	301
NC10	4.2250	1.4740	4.2247	1.4740	5344

Table 6 Numerical results of Algorithm 2 (DCU/CCF)

Problem	x_{DCU}	f_{DCU}	$N_f, N_{f'}$	N_g	$N_{g'}$	$N_{\text{total}}^{\text{DCU}}$	SPEEDUP
NC01	1.0574	−7.6129	7	17	11	42	0.8571
NC02	1.0188	5.4539	5	44	40	94	2.6383
NC03	−5.9922	−2.9468	12	48	37	109	0.7706
NC04	2.4594	2.8408	23	88	66	288	5.0660
NC05	9.2846	−1.2748	14	26	13	93	4.3226
NC06	2.3240	−1.6852	7	50	44	158	1.4430
NC07	−0.7747	−0.3301	32	54	23	195	4.0718
NC08	−1.1274	−6.6006	18	68	51	291	1.8419
NC09	4.0000	1.9222	13	42	30	182	1.6538
NC10	4.2262	1.4740	76	127	52	585	9.1350
Average							3.1800

(1) The columns x_{DCU} and f_{DCU} represent the global optimum and its objective function values found by Algorithm 2, respectively.
(2) The columns N_f, $N_{f'}$, N_g, and $N_{g'}$ represent the number of function evaluations of f, f', g, and g', respectively.
(3) The column $N_{\text{total}}^{\text{DCU}}$ represents the total number of function evaluations.
(a)$N_f + N_{f'} + N_g + N_{g'}$, for problems with one constraint;

(b)$N_f + N_{f'} + 2N_g + N_{g'}$, for problems with two constraints;
(c)$N_f + N_{f'} + 3N_g + N_{g'}$, for problems with three constraints.

(4) The column 'SPEEDUP' represents the index for comparison with IBBA:

$$\text{SPEEDUP} = \frac{\mathrm{N}_{\text{total}}^{\text{IBBA}}}{\mathrm{N}_{\text{total}}^{\text{DCU}}}.$$

For the number of function evaluations, most problems, except problems NC01 and NC03, have 'SPEEDUP' values greater than 1.

7 Conclusion

In this paper, new global optimization technique has been proposed for single-variable, twice-differentiable problems based on the difference of convex underestimator and the convex cut function. For unconstrained problems, convergence of the basic algorithm was proved, and an upper bound for the asymptotic error constant was suggested. Numerical tests show that the upper bound is at most twice as large as the actual constant. It was also demonstrated that the algorithm requires less computational load compared with BC and SA. For constrained problems, algorithm 2 was employed and exhibited quadratic convergence to locate a zero of constraint. By the aid of a convex cut function, constrained optimization algorithm guarantees determination of a global optimum. The numerical experiments indicate that the proposed algorithm competes with another covering method, IBBA, which uses the Lipschitz constant. The proposed algorithm obtains the optimum values of the objective function and decision variable for all problems and requires a lower number of total function evaluations for most problems.

References

Adjiman, C.S., Androulakis, I.P., Maranas, C.D., Floudas, C.A.: A global optimization method, αBB, for process design. Comp. Chem. Engng. Suppl. **20**, S419–S424 (1996)

Adjiman, C.S., Androulakis, I.P., Floudas, C.A.: Global optimization of mixed-integer nonlinear problems. AIChE J. **46**, 1769–1797 (2000)

Adjiman, C.S., Dallwig, S., Floudas, C.A.: A global optimization method, αBB, for general twice-differentiable constrained NLPs—I. Theoretical advances. Comp. Chem. Engng. **22**, 1137–1158 (1998a)

Adjiman, C.S., Dallwig, S., Floudas, C.A.: A global optimization method, αBB, for general twice-differentiable constrained NLPs—II. Implementation and computational results. Comp. Chem. Engng. **22**, 1159–1179 (1998b)

Adjiman, C.S., Floudas, C.A.: Rigorous convex underestimators for general twice–differentiable problems. J. Glob. Optim. **9**, 23–40 (1996)

Akrotirianakis, I.G., Floudas, C.A.: A new class of improved convex underestimators for twice continuously differentiable constrained NLPs. J. Glob. Optim. **30**, 367–390 (2004a)

Akrotirianakis, I.G., Floudas, C.A.: Computational experience with a new class of convex underestimators: box-constrained NLP problems. J. Glob. Optim. **29**, 249–264 (2004b)

Androulakis, I.P., Maranas, C.D., Floudas, C.A.: αBB: a global optimization method for general constrained nonconvex problems, J. Glob. Optim. **7**, 337–363 (1995)

Basso, P.: Iterative method for localization of the global maximum. SIAM J. Num. Anal. **19**, 781–792 (1982)

Breiman, L., Culter, A.: A derterministic alogorithm for global optimization. Math. Program. **58**, 179–199 (1993)

Byrne, R.P., Bogle, I.D.L.: Global optimization of constrained non-convex programs using reformulation and interval analysis. Comp. Chem. Engng. **23**, 1341–1350 (1999)

Caratzoulas, S., Floudas, C.A.: Trigonometric convex underestimator for the base functions in fourier space. J. Optim Theory Appl. **124**, 339–362 (2005)

Elwakeil, O.A., Arora, R.S.: Two algorithms for global optimization of general NLP problems, Int. J. Num. Methods Eng. **39**, 3305–3325 (1996)

Esposito, W.R., Floudas, C.A.: Global optimization for the parameter estimation of differential algebraic systems. Ind. Chem. Engng. Res. **39**, 1291–1310 (2000)

Famularo, D., Sergeyev, YA.D., Pugliese, P.: Test problems for Lipschitz univariate global opitmization with multiextremal constraints. In: Dzemyda, G., Saltenis, V., Zilinskas, A. (eds.) Stochastic and Global Optimization. Kluwer Academic Publishers, The Netherlands (2001)

Floudas, C.A.: Deterministic Global Optimization: Theory, Methods and Application. Kluwer Academic Publishers, The Netherlands (2000a)

Floudas, C.A.: Global optimization in design and control of chemical process systems, J. Process Control **10**, 125–134 (2000b)

Floudas, C.A., Akrotirianakis, I.G., Caratzoulas, S., Meyer, C.A., Kallrath, J.: Global optimization in the 21st century: advances and challenges. Comp. Chem. Engng. **29**, 1185–1202 (2005)

Gergel, V.P., Sergeyev, Y.D.: Sequential and parallel algorithms for global minimizing functions with Lipschitzian dervatives, Comput. Math. Appl. **37**, 163–179 (1999)

Hansen, P., Jaumard, B.: Lipschitz optimization. In: Horst, R., Pardalos, M.P. (eds.) Handbook of Global Optimization. Kluwer Academic Publishers, The Netherlands (1995)

Hansen, P., Jaumard, B., Lu, S.-H.: Global optimization of univariate Lipschitz functions: II. New algorithms and computational comparison. Math. program. **55**, 273–292 (1992a)

Hansen, P., Jaumard, B., Lu, S.-H.: On the use of estimates of the Lipschitz constant in global optimization. J. Optim. Theory Appl. **75**, 195–200 (1992b)

Hertz, D., Adjiman, C.S., Floudas, C.A.: Two results on bounding the roots of interval polynomials. J. Comp. Chem. Engng. **23**, 1333–1339 (1999)

Horst, R., Tuy, H.: Global Optimization: Deterministic Approaches. Springer-Verlag, Berlin (1996)

Ichida, K.: Constrained optimization using interval analysis. Comp. Ind. Engng. **31**, 933–937 (1996)

Jansson, C.: Quasiconvex relaxations based on interval arithmetic. Linear Algebra Appl. **324**, 27–53 (2001)

Kearfott, B.R.: A fortran 90 environment for research and prototyping of enclosure algorithms for nonlinear eqautions and global optimization. ACM Trans. Math. Softw. **21**, 63–78 (1995)

Kim, Y., Lee, T.: Acceleration of αBB global optimization algorithm using quadratic and linear underestimator. ESCAPE-11 Suppl. Proc. 35–40 (2001)

MacLagan, D., Sturge, T., Baritompa, W.P.: Equivalent methods for global optimization. In: Floudas, C.A., Pardalos, P.M. (eds.) State of the Art in Global Optimization. Kluwer Academic Publishers, Dordrecht (1996)

Maranas, C.D., Floudas, C.A.: A global optimization approach for Lennard-Jones microclusters. J. Chem. Phys. **97**, 7667–7677 (1992)

Maranas, C.D., Floudas, C.A.: A deterministic global optimization approach for molecular structure determination, J. Chem. Phys. **100**, 1247–1261 (1994)

Mayne, D.Q., Polak, E.: Outer approximation algorithm for non-differentiable optimization problems, J. Optim. Theory Appl. **42**, 19–30 (1984)

Meyer, C.A., Floudas, C.A.: Convex hull of trilinear monomials with mixed-sign domains. J. Glob. Optim. **29**, 125–155 (2004)

Meyer, C.A., Floudas, C.A.: Convex underestimation of twice continuously differentiable functions by piecewise quadratic perturbation: spline αBB underestimators. J. Glob. Optim. **32**, 221–258 (2005)

Meyer, C.A., Floudas, C.A., Neumaier, A.: Global optimization with nonfractable constraints. Ind. Chem. Engng. Res. **41**, 6413–6424 (2002)

Pijavskii, S.A.: An algorithm for finding the absolute extremum of a function. USSR Comput. Math. Math. Phys. **12**, 57–67 (1972)

Ryoo, H.S., Sahinidis, N.V.: Global optimization of nonconvex NLPs and MINLPs with applications in process design. Comp. Chem. Engng. **19**, 551–566 (1995)

Sergeyev, Y.D.: Global one-dimensional optimization using smooth auxiliary functions. Math. Program. **81**, 127–146 (1998)

Sergeyev, Y.D., Famularo, D., Pugliese, P.: Index branch-and-bound algorithm for Lipcshitz univariate global optimization with multiextremal constraints. J. Glob. Optim. **21**, 317–341 (2001)

Sergeyev, Y.D., Famularo, D., Pugliese, P.: Index information algorithm with local tuning for solving multidimensional global optimization problems with multiextremal constraints. Math. Program. **96**, 489–512 (2003)

Smith, E.M.B., Pantelides, C.C.: Global optimization of nonconvex MINLPs. Comp. Chem. Engng. **21S**, S791–S796 (1997)
Wang, T., Wah, B.W.: Handling inequality constraints in continuous nonlinear global optimization. Integr. Design Process Technol. 267–274 (1996)
Zamora, J.M., Grossmann, I.E.: A global MINLP optimization algorithm for the synthesis of heat exchanger networks with no stream splits. Comp. Chem. Engng. **22**, 367–384 (1998)