Tighter McCormick Relaxations through Subgradient Propagation

Jaromił Najman · Alexander Mitsos*

RWTH Aachen University
AVT - Aachener Verfahrenstechnik
Process Systems Engineering
Forckenbeckstraße 51
D - 52074 Aachen

Abstract Tight convex and concave relaxations are of high importance in the field of deterministic global optimization. We present a method to tighten relaxations obtained by the McCormick technique. We use the McCormick subgradient propagation (Mitsos et al., SIAM J. Optim., 2009) to construct simple affine under- and overestimators of each factor of the original factorable function. Then, we minimize and maximize these affine relaxations in order to obtain possibly improved range bounds for every factor resulting in possibly tighter final McCormick relaxations. We discuss the method and its limitations, in particular the lack of guarantee for improvement. Subsequently, we provide numerical results for benchmark cases found in the COCONUT library and case studies presented in previous works, where the McCormick technique appears to be advantageous, and discuss computational efficiency. We see that the presented algorithm provides a significant improvement in tightness and decrease in computational time in many cases.

1 Introduction

Methods based on branch-and-bound (B&B) [30] are the state-of-the art algorithms in the field of deterministic global optimization. In general, B&B methods rely on favorable convergence order of the underlying convex and concave relaxations of all functions involved in a given global optimization problem in order to avoid the so-called cluster effect [15, 20, 21, 22, 50]. Domain and range reduction techniques are employed within B&B algorithms in order to further increase the quality of the underlying relaxations. These techniques are not necessary to guarantee convergence of the B&B algorithm, however, they are able to drastically speed up convergence.

Relaxation techniques based on interval arithmetic [29, 37] describe a general way to obtain valid over- and underestimations of a multivariate function $f : Z \to \mathbb{R}$. It is a well-known fact that the simplest interval-based method, called natural interval extensions, often provides very loose estimators. Thus, efforts have been made to improve tightness by introducing rigorous arithmetic extensions such as affine reformulations [12, 14, 34] and improved Taylor models [6, 40].

The method of constructing valid convex and concave relaxations of a continuous factorable function, given by a finite recursion of addition, multiplication and composition via propagation of valid factors, e.g., $F_1 \circ f_1 + F_2 \circ f_2 \cdot F_3 \circ f_3$, was presented by McCormick [25, 26] and extended to multivariate outer functions

Corresponding author: *A. Mitsos E-mail: amitsos@alum.mit.edu

F_i in [48]. McCormick's idea was used in the development of the well-known auxiliary variable method (AVM) [45, 46, 47] used in state-of-the-art global optimization solvers such as BARON [47] and ANTIGONE [27].

In order to further improve the tightness of the relaxations constructed with the AVM, many bound-tightening procedures are used such as Optimality-Based Bound Tightening [23], where additional optimization problems are solved in order to tighten variable bounds; bound propagation techniques [10], where information on a constraint is used to possibly tighten the bounds of a different constraint and finally the variables involved in both constraints; probing [46], where valid constraints are derived from non-active constraints and more. The recent article by Puranik and Sahinidis [36] provides a thorough overview of the field of tightening techniques for AVM. Most of the techniques applicable to the AVM are (at least theoretically) directly applicable to the relaxations obtained via the McCormick technique. Still, there are almost no algorithms developed directly for the improvement of relaxations obtained by the McCormick method. Wechsung et al. [51] present an algorithm for constraint propagation using McCormick relaxations resulting in a reduced variable domain and tighter final McCormick relaxations. They reverse the operations starting with pre-computed McCormick relaxations of a given factorable function g and traverse the factors of g backwards in order to tighten the set of feasible points. Herein, we present a different idea with the same goal of improving the final resulting McCormick relaxations. The presented algorithm uses subgradient propagation [28] in order to possibly improve interval range bounds in each factor of g resulting in a tighter final relaxation.

The remainder of the manuscript is structured as follows. In Section 2, we provide basic definitions and notation used throughout the article. We present the algorithm in Section 3 supported by an example and discuss similarities and differences to methods used in AVM and the limitations of the presented method. Subsequently, we present numerical results in Section 4 and examine different adjustments of the presented algorithm. Section 5 concludes the work.

2 Basic definitions

In the following, if not stated otherwise, we consider a continuous function $f : Z \to \mathbb{R}$ with $Z \in \mathbb{IR}^n$, where $\mathbb{IR}$ denotes the set of closed bounded intervals of $\mathbb{R}$. $Z \in \mathbb{IR}^n$, also called *box*, is defined as $Z \equiv [\mathbf{z}^L, \mathbf{z}^U] = [z_1^L, z_1^U] \times \cdots \times [z_n^L, z_n^U]$ with $\mathbf{z}^L, \mathbf{z}^U \in \mathbb{R}^n$ where the superscripts L and U always denote a lower and upper bound, respectively. We denote the image of f over Z by $f(Z) \in \mathbb{IR}$. We denote the estimation of the range bounds of f on Z with the use of natural interval extensions by $I_{f,nat} \supset f(Z)$ and the exact bounds by $I_{f,e} = f(Z)$.

We call a convex function $f^{cv} : Z \to \mathbb{R}$ a convex relaxation (or convex underestimator) of f on Z if $f^{cv}(\mathbf{z}) \leqslant f(\mathbf{z})$ for every $\mathbf{z} \in Z$. Similarly, we call a concave function $f^{cc} : Z \to \mathbb{R}$ a concave relaxation (or concave overestimator) of f on Z if $f^{cc}(\mathbf{z}) \geqslant f(\mathbf{z})$ for every $\mathbf{z} \in Z$. We call the tightest convex and concave relaxations of f the convex and concave envelopes f_e^{cv}, f_e^{cc} of f on Z, respectively, i.e., it holds $f^{cv}(\mathbf{z}) \leqslant f_e^{cv}(\mathbf{z}) \leqslant f(\mathbf{z})$ and $f(\mathbf{z}) \leqslant f_e^{cc}(\mathbf{z}) \leqslant f^{cc}(\mathbf{z})$ for all $\mathbf{z} \in Z$ and all convex relaxations f^{cv} and concave relaxations f^{cc} of f on Z, respectively.

For a convex and concave function $f^{cv}, f^{cc} : Z \to \mathbb{R}$, we call $\mathbf{s}^{cv}(\bar{\mathbf{z}}), \mathbf{s}^{cc}(\bar{\mathbf{z}}) \in$

$\mathbb{R}^n$ a convex and a concave subgradient of f^{cv}, f^{cc} at $\bar{\mathbf{z}} \in Z$, respectively, if

$$f^{cv}(\mathbf{z}) \geqslant f^{cv}(\bar{\mathbf{z}}) + (\mathbf{s}^{cv}(\bar{\mathbf{z}}))^T(\mathbf{z}-\bar{\mathbf{z}}), \quad \forall \mathbf{z} \in Z, \tag{A1}$$

$$f^{cc}(\mathbf{z}) \leqslant f^{cc}(\bar{\mathbf{z}}) + (\mathbf{s}^{cc}(\bar{\mathbf{z}}))^T(\mathbf{z}-\bar{\mathbf{z}}), \quad \forall \mathbf{z} \in Z. \tag{A2}$$

We denote the affine functions on the right-hand side of inequalities (A1), (A2) constructed with the convex and concave subgradient $\mathbf{s}^{cv}(\bar{\mathbf{z}}), \mathbf{s}^{cc}(\bar{\mathbf{z}})$ by $f^{cv,sub}(\bar{\mathbf{z}}, \mathbf{z})$ and $f^{cc,sub}(\bar{\mathbf{z}}, \mathbf{z})$, respectively. Note that $f^{cv,sub}$ and $f^{cc,sub}$ are valid under- and overestimators of f on Z, respectively.

2.1 McCormick relaxations and subgradient propagation

We will make use of McCormick propagation rules originally developed by McCormick [25] and extended to multivariate compositions of functions by Tsoukalas and Mitsos [48]. Rules for the propagation of subgradients for the unviariate composition McCormick rule are given in [28] and of multivariate composition rule in [48]. The rules for the construction of McCormick relaxations of binary sum, binary product and univariate composition can be found in, e.g., Propositions 2, 3 and Theorem 5 in [5], Propositions 2.6, 2.7 and Theorem 2.8 in [28] or Section 3 in [25]. The corresponding subgradient propagation rules can be found in Proposition 2.9 and Theorems 3.2 and 3.3 in [28]. The rule for the construction of McCormick relaxations with multivariate outer functions can be found in Theorem 2 in [48] and the corresponding subgradient propagation rule in Theorem 4 of [48].

2.2 Computational Graph

We assume that a directed acyclic graph (DAG) representation $G = (\mathbb{F}, E)$, described in, e.g., Sections 2 and 3 in [41], of a (multivariate) factorable function $g : Z \to \mathbb{R}$ with $Z \in \mathbb{IR}^n$ is given. $\mathbb{F}$ is the set of vertices, which we call factors herein, consisting of operations and independent variables, and E the set of edges connecting the factors. For non-commutative operations, e.g., subtraction, the correct order of the previous factors is saved and shown in Fig. 1 from left to right, i.e., the left child of the '$-$' operation is on the left side of the minus sign and the right child is on the right side of it. We assume that for each factor $f_j \in \mathbb{F}, j \in \{1, \ldots, |\mathbb{F}|\}$ convex $f_j^{cv}(\bar{\mathbf{z}})$ and concave $f_j^{cc}(\bar{\mathbf{z}})$ relaxations, the corresponding convex $\mathbf{s}_j^{cv}(\bar{\mathbf{z}})$ and concave $\mathbf{s}_j^{cc}(\bar{\mathbf{z}})$ subgradients at a point $\bar{\mathbf{z}} \in Z$ and valid upper f_j^U and lower f_j^L bounds on the range of f over Z are calculated, see Example 4.2 and Fig. 4.2 in [28]. The relaxations and subgradients are calculated by the McCormick rules. The upper f_j^U and lower f_j^L bounds on the range of f over Z are obtained via natural interval extension ([29, 37]) throughout this article, i.e., $I_{j,nat} = [f_j^L, f_j^U]$. In order to evaluate g, its relaxations and its subgradients at a point $\bar{\mathbf{z}} \in Z$ through G, we assume that the corresponding DAG is traversed in a reversed-level-order, i.e., starting at the independent variables $z_i,\ i \in \{1, \ldots, n\}$ and working through all factors up to the root given as $g(\bar{\mathbf{z}})$.

Example 1 Consider the function $g(z) = (z - z^2)(z^3 - \exp(z))$ on $Z = [-0.5, 1]$. It consists of 7 factors, namely the independent variable z and the 6 operations $\hat{}2, \hat{}3, \exp, -, -$ and $\times$. The corresponding computational graph is shown in Fig. 1.

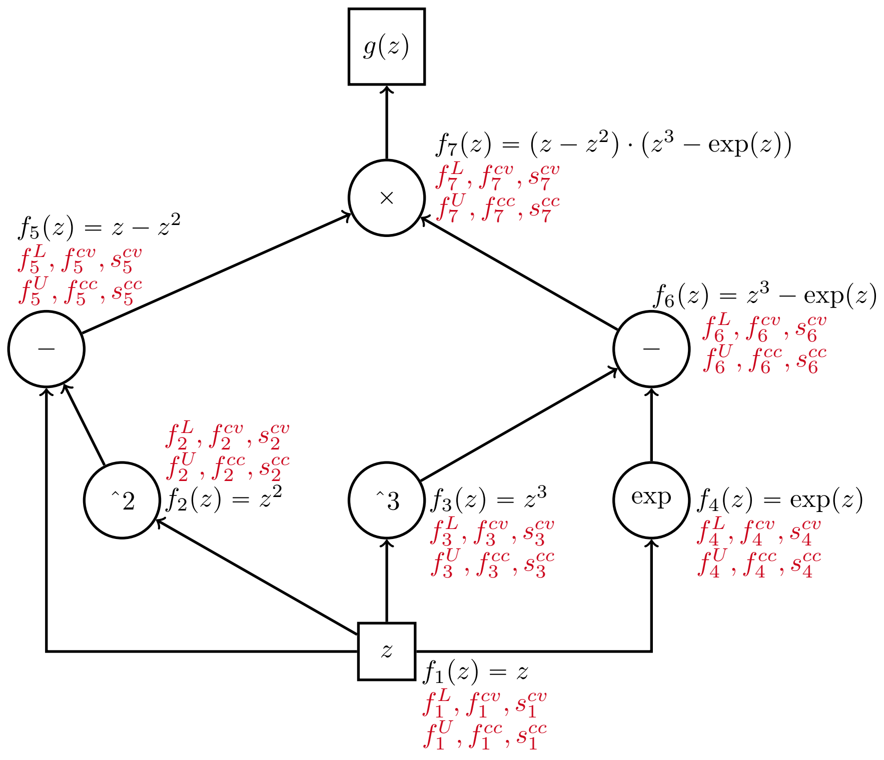

$g(z)$
$f_7(z) = (z - z^2) \cdot (z^3 - \exp(z))$
$f_7^L, f_7^{cv}, s_7^{cv}$
$f_7^U, f_7^{cc}, s_7^{cc}$
$\times$
$f_5(z) = z - z^2$
$f_5^L, f_5^{cv}, s_5^{cv}$
$f_5^U, f_5^{cc}, s_5^{cc}$
$-$
$f_6(z) = z^3 - \exp(z)$
$f_6^L, f_6^{cv}, s_6^{cv}$
$f_6^U, f_6^{cc}, s_6^{cc}$
$-$
$f_2^L, f_2^{cv}, s_2^{cv}$
$f_2^U, f_2^{cc}, s_2^{cc}$
^2
$f_2(z) = z^2$
^3
$f_3(z) = z^3$
$f_3^L, f_3^{cv}, s_3^{cv}$
$f_3^U, f_3^{cc}, s_3^{cc}$
exp
$f_4(z) = \exp(z)$
$f_4^L, f_4^{cv}, s_4^{cv}$
$f_4^U, f_4^{cc}, s_4^{cc}$
z
$f_1(z) = z$
$f_1^L, f_1^{cv}, s_1^{cv}$
$f_1^U, f_1^{cc}, s_1^{cc}$

Fig. 1: Computational graph for $g(z) = (z - z^2) \cdot (z^3 - \exp(z))$ on $Z = [-0.5, 1]$.

3 Algorithm for tighter McCormick relaxations

3.1 Basic idea

For a nonlinear factorable function given by a finite recursion of addition, multiplication and composition, $g = f_1 \circ f_2 + f_3 \circ f_4 \cdot f_5 \circ f_6$, there are several bound and domain tightening techniques and ideas, found in, e.g., [9, 13, 17, 18, 35] just to name a few and more can be found in the recent article by Puranik and Sahinidis [36]. Many tightening methods use information on constraints within a given problem in order to tighten variable bounds, e.g., [9, 17, 44], while other methods use optimality conditions, reduced costs of variables, and dual multipliers of constraints of the given problem to obtain a tighter relaxation, e.g., [35, 39, 46]. We present an algorithm which uses information on McCormick relaxations and subgradients of each factor of a particular function g within a given optimization problem to possibly improve the resulting final McCormick relaxations of g. In Section 2.3 of [33], we have presented that using tighter range bounds for each factor of a McCormick relaxation results in tighter relaxations. Herein, we present an idea for obtaining tighter McCormick relaxations with the use of subgradients for McCormick relaxations, [28] (implemented within MC++(v2.0)[11]). The presented algorithm is not guaranteed to improve the final McCormick relaxations making it a heuristic. We first give the basic idea followed by an example and then formalize the algorithm.

Typically, when calculating McCormick relaxations, natural interval extensions are used for the computation of valid interval bounds for the range of each factor of g. This is also the case within MC++(v2.0)[11]. It is possible to use different interval extensions, e.g., the standard centered form or Taylor forms (Sections 2.2 and 3.7 in [37]). We show that the presented algorithm can still improve the final McCormick relaxations for more sophisticated interval arithmetic and briefly discuss results in Section 3.2.2. By solving $\min\limits_{\mathbf{z}\in Z} f_j^{cv}(\mathbf{z})$ and $\max\limits_{\mathbf{z}\in Z} f_j^{cc}(\mathbf{z})$ for each factor $f_j, j \in \{1,\ldots,|\mathbb{F}|\}$ of a factorable function g, where f_j^{cv}, f_j^{cc} are McCormick relaxations of f_j, we can obtain valid and possibly tighter range bounds for each factor f_j. We can achieve even tighter bounds by solving $\min\limits_{\mathbf{z}\in Z} f_j(\mathbf{z})$ and $\max\limits_{\mathbf{z}\in Z} f_j(\mathbf{z})$ resulting in a possible improvement of McCormick relaxations of g (see Example 2 and Example 5 in [33]). However, the number of factors in a factorable function can be very large leading to a high computational time. Thus, we want to find a good trade-off between tightness of range bounds of each factor and computational time needed. We could approximately solve $\min\limits_{\mathbf{z}\in Z} f_j(\mathbf{z})$ and $\max\limits_{\mathbf{z}\in Z} f_j(\mathbf{z})$ using linear or higher order approximations of f_j in order to simplify the optimization problem but this does not guarantee valid bounds. We could as well approximately solve $\min\limits_{\mathbf{z}\in Z} f_j^{cv}(\mathbf{z})$ and $\max\limits_{\mathbf{z}\in Z} f_j^{cc}(\mathbf{z})$ by the use of a solution method for convex (nonsmooth) problems, e.g., bundle-methods ([3, 4]), and allow only a small number of iteration steps providing valid but possibly extremely loose bounds. When computing McCormick relaxations of a factorable function g, each factor has information about its range bounds, convex and concave McCormick relaxations and its convex and concave subgradients. In this article, we use this information in order to solve $\min\limits_{\mathbf{z}\in Z} f_j^{cv}(\mathbf{z})$ and $\max\limits_{\mathbf{z}\in Z} f_j^{cc}(\mathbf{z})$ approximately by solving the simple linear box-constrained problems $\min\limits_{\mathbf{z}\in Z} f_j^{cv,sub}(\bar{\mathbf{z}},\mathbf{z})$ and $\max\limits_{\mathbf{z}\in Z} f_j^{cc,sub}(\bar{\mathbf{z}},\mathbf{z})$ for every factor $f_j, j \in \{1,\ldots,|\mathbb{F}|\}$ resulting in possibly improved range bounds f_j^L, f_j^U and finally tighter convex and concave McCormick relaxations of g.

In Example 4.4 of [28], simple affine relaxations are constructed by using the propagated subgradient at a point $\bar{\mathbf{z}} \in Z$ to construct the affine relaxations $g^{cv,sub}(\bar{\mathbf{z}},\mathbf{z}), g^{cc,sub}(\bar{\mathbf{z}},\mathbf{z})$ of the original function g. The lower bound obtained by evaluating the affine functions at their minimum and maximum, respectively, can result in tighter bounds than the underlying (natural) interval extensions, see Fig. 4.4 in [28]. We can exploit this property of the affine estimators when constructing McCormick relaxations by computing the subgradients at a point in the domain, e.g., the middle-point, in each factor f_j of the factorable function g and checking if we can improve the corresponding range bounds for the current factor, i.e., we approximately solve $\min\limits_{\mathbf{z}\in Z} f_j^{cv}(\mathbf{z})$ and $\max\limits_{\mathbf{z}\in Z} f_j^{cc}(\mathbf{z})$ by solving $\min\limits_{\mathbf{z}\in Z} f_j^{cv,sub}(\bar{\mathbf{z}},\mathbf{z})$ and $\max\limits_{\mathbf{z}\in Z} f_j^{cc,sub}(\bar{\mathbf{z}},\mathbf{z})$ for every $j \in \{1,\ldots,|\mathbb{F}|\}$. We obtain the corresponding values $f_{j,alg}^L, f_{j,alg}^U$ and check if $f_{i,new}^L > f_j^L, f_{j,alg}^U < f_j^U$, i.e., if we can improve the bounds on the range of f_j for each $j \in \{1,\ldots,|\mathbb{F}|\}$. Note that it is not guaranteed that the bounds $f_{j,alg}^L, f_{j,alg}^U$ are better than the bounds f_j^L, f_j^U obtained through natural interval extensions. Subsequently, we can compute a next linearization point and repeat the computations if desired. Note that since we cannot provide any guarantee on the new bounds $f_{j,alg}^L, f_{j,alg}^U$, it can take many re-computations in order to achieve an improvement for the range bounds f_j^L, f_j^U of a factor f_j and thus, a maximal number of iterations should be predefined.

3.1.1 Connection to existing ideas

The idea can be intuitively described as a specific application of subgradient bundle methods ([3, 4]). We use subgradients to approximate the possibly non-smooth convex/concave relaxations of a given factor f_j of the original function g representing our *bundle*. We then construct a linear approximation of f_j and check if the linearization provides a better range bound than the natural interval extension. If a maximum number of iterations is not reached, we use the subgradient information in order to determine a new point. In contract to a bundle method, we always work with only one subgradient, i.e., bundle of size 1, making the application easier. We discuss an alternative of the presented method in Section 3.6. It is also be possible to describe this idea as a modified Sandwich algorithm (4.2 in [46]) where polyhedral approximations are constructed for convex functions. However, the Sandwich algorithm does not work with propagated subgradients but rather with the differentials of particular convex functions. Moreover, we try to avoid computing subgradients at all corners, in contrast to what is done in Figures 4.1-4.8 in [46], since the dimension of $\mathbf{z}$ can be too large to make this algorithm applicable in each factor of g. The idea of the method is presented in the next example.

3.2 Illustrative examples

3.2.1 Natural interval extensions

Example 2 Consider again the function $g(z) = (z - z^2)(z^3 - \exp(-z))$ on $Z = [-0.5, 1]$, Fig. 3, and consider the particular three factors $f_5(z) = z - z^2$, $f_6(z) = z^3 - \exp(z)$ and $f_7(z) = f_5(z) \cdot f_6(z)$. For factors f_1, f_2, f_3, f_4, envelopes are known and natural interval extension provide exact range bounds. The convex and concave McCormick relaxations provide envelopes for f_5 on Z given as

$$f_5^{cv}(z) = 0.5z - 0.5 \text{ and } f_5^{cc}(z) = z - z^2.$$

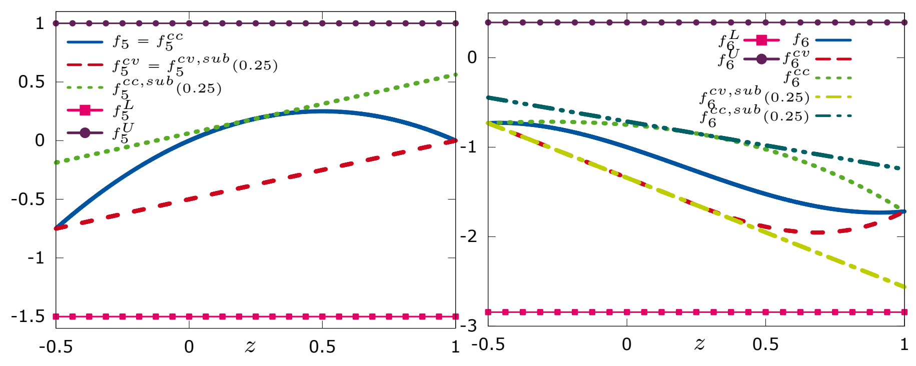
The natural interval extensions are not exact for the range of f_5 providing $I_{5,nat} = [-1.5, 1]$ while the exact range is given as $I_{5,e} = [-0.75, 0.25]$. The convex and concave subgradient of f_5^{cv} and f_5^{cc}, respectively, at the middle point 0.25 of Z are $s_5^{cv}(0.25) = 0.5$ and $s_5^{cc}(0.25) = 0.5$, respectively. We construct the corresponding affine functions

$$\begin{aligned} f_5^{cv,sub}(0.25, z) &= f_5^{cv}(0.25) + s_5^{cv}(0.25)(z - 0.25) \text{ and} \\ f_5^{cc,sub}(0.25, z) &= f_5^{cc}(0.25) + s_5^{cc}(0.25)(z - 0.25). \end{aligned}$$

Next, we evaluate the affine functions at their respective minimum and maximum in order to obtain

$$\begin{aligned} \min_{z\in[-0.5,1]} f_5^{cv,sub}(0.25, z) &= f_5^{cv,sub}(0.25, -0.5) = -0.75 \text{ and} \\ \max_{z\in[-0.5,1]} f_5^{cc,sub}(0.25, z) &= f_5^{cc,sub}(0.25, 1) = 0.5625. \end{aligned} \tag{1}$$

With (1) we can improve the natural interval extensions range bounds $I_{5,nat}$ from $[-1.5, 1]$ to $I_{5,alg} = [-0.75, 0.5625]$ for factor f_5. The factor f_5 together



(a) (b)

Fig. 2: Example 2. **(a)** Factor $f_5(z) = z - z^2$ with its convex and concave McCormick relaxations, natural interval extension estimators f_5^L, f_5^U for the range of f_5 on $Z = [-0.5, 1]$ and affine under- and overestimators constructed with the use of subgradients at the middle point 0.25. The affine underestimator equals the convex envelope of f_5.
(b) Factor $f_6(z) = z^3 - \exp(z)$ with its convex and concave McCormick relaxations, natural interval extension estimators f_6^L, f_6^U for the range of f_6 on $Z = [-0.5, 1]$ and affine under- and overestimators constructed with the use of subgradients at the middle point 0.25.

with its convex and concave McCormick relaxations (constructed with natural interval bounds), natural interval bounds f_5^L, f_5^U and the affine functions can be seen in Fig. 2a. This procedure can be rerun for a different point in order to possibly improve the interval bounds even further but to keep this example simple, we do only one iteration. Note that it is possible to rerun the procedure for the same point if we improve a factor before, since the relaxations and the corresponding subgradients change.

Next, we compute improved range bounds for the factor $f_6(z) = z^3 - \exp(z)$. The convex and concave McCormick relaxations of f_6 on Z (using the supplementary material of [42]) are given as

$$f_6^{cv}(z) = \begin{cases} -0.125 - \exp(-0.5) + \left(0.1875 - \frac{\exp(-0.5)-\exp(1)}{-0.5-1}\right)(z+0.5) & \text{, for } z \leqslant 0.25 \\ z^3 - \exp(-0.5) - \frac{\exp(-0.5)-\exp(1)}{-0.5-1}(z+0.5) & \text{, else} \end{cases}$$
$$f_6^{cc}(z) = 0.25 + 0.75z - \exp(z).$$

The natural interval extensions provide $I_{6,nat}=[-\exp(1)-0.125, 1-\exp(-0.5)] \approx [-2.843, 0.393]$, while the exact range is given as $I_{6,e} \approx [-1.73, -0.728]$. The convex and concave subgradients of f_6^{cv} and f_6^{cc}, respectively, at the middle point 0.25 of Z are $s_6^{cv}(0.25) = 0.1875 - \frac{\exp(-0.5)-\exp(1)}{-0.5-1}$ and $s_6^{cc}(0.25) = 0.75 - \exp(0.25)$. We construct the corresponding affine functions

$$f_6^{cv,sub}(0.25, z) = f_6^{cv}(0.25) + s_6^{cv}(0.25)(z - 0.25) \text{ and}$$
$$f_6^{cc,sub}(0.25, z) = f_6^{cc}(0.25) + s_6^{cc}(0.25)(z - 0.25).$$

Then, we evaluate the affine functions at their respective minimum and maxi-

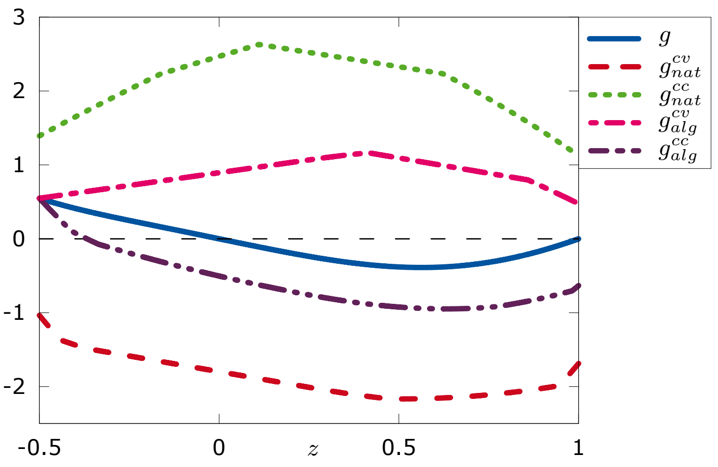


Fig. 3: Example 2. Function $g(z) = (z - z^2)(z^3 - \exp(z))$ on $Z = [-0.5, 1]$ together with its convex and concave McCormick relaxations $g^{cv}_{nat}, g^{cc}_{nat}$ constructed with natural interval extensions and the convex and concave McCormick relaxation $g^{cv}_{alg}, g^{cc}_{alg}$ constructed using the range bounds computed Algorithm 1 with only one iteration at each factor.

mum in order to obtain

$$\begin{aligned}
&\min_{z\in[-0.5,1]} f_6^{cv,sub}(0.25, z) = f_6^{cv,sub}(0.25, 1) \approx -2.562 \text{ and} \\
&\max_{z\in[-0.5,1]} f_6^{cc,sub}(0.25, z) = f_6^{cc,sub}(0.25, -0.5) \approx -0.446.
\end{aligned} \tag{2}$$

With (2) we can improve the natural interval extensions range bounds $I_{6,nat}$ from $\approx [-2.843, 0.393]$ to $I_{6,alg} \approx [-2.562, -0.446]$ for factor f_6. Factor f_6 together with its convex and concave McCormick relaxations (constructed with natural interval bounds), natural interval bounds f_6^L, f_6^U and the affine functions can be seen in Fig. 2b. Once again, this procedure can be rerun for a different point in order to possibly improve the interval bounds even further but to keep this example simple, we do only one iteration.

We can now construct the envelope for the bilinear product $f_7 = f_5 f_6$ on the improved intervals $I_{5,alg} \times I_{6,alg}$ and finally the convex and concave McCormick relaxations for g. Figure 3 shows function g together with its convex and concave McCormick relaxations constructed with the simple intervals $I_{5,nat}, I_{6,nat}$ denoted as $g^{cv}_{nat}, g^{cc}_{nat}$ and two improved McCormick relaxations constructed with the new intervals $I_{5,alg}, I_{6,alg}$ denoted as $g^{cv}_{alg}, g^{cc}_{alg}$. The proposed algorithm drastically improves the relaxations.

The procedure for obtaining improved lower and upper bounds for a factor of g is formally given in Section 3.3 and discussed afterwards.

3.2.2 Advanced interval extensions

Although natural interval extensions are very common in optimization for their simplicity, robustness and extremely low computational times, the usage of more sophisticated interval arithmetic is often advisable. If better interval extensions

are used for the construction of McCormick relaxations, the resulting under- and overestimators may be much tighter than relaxations constructed with natural interval extensions, cf. Example 5 in [33]. Still, even the more sophisticated interval extensions do not guarantee that the resulting range bounds are exact. This leaves room for improvement of the range bounds by the presented method.
Example 3 Consider $g(z) = (\log(z+1) - z^2)(\log(z+1) - \exp(z-0.5))$ on $Z = [-0.5, 1]$, Fig. 5, and consider the particular three factors $f_9(z) = \log(z+1) - z^2$, $f_{10}(z) = \log(z+1) - \exp(z-0.5)$ and $f_{11}(z) = f_9(z) \cdot f_{10}(z)$. For the other factors $z, 1, 0.5, z+1, z-0.5, z^2, \log(z+1), \exp(z-0.5)$, envelopes are known and simple interval arithmetic provide exact range bounds. In this example, we use the second order Taylor form interval extensions (Section 3.7 in [37]) instead of natural interval extension to show that the presented algorithm can provide tighter McCormick relaxations even if more advanced interval arithmetic are used for the construction of McCormick relaxations. In particular, we compute the range bounds for a twice differentiable function $h : Z \to \mathbb{R}$ by calculating $I_{h,T} = [h_T^L, h_T^U] = h(c) + h'(c)(Z-c) + \frac{h''(Z)}{2}(Z-c)^2$, where c is the middle point of Z, h', h'' are the first and second derivatives of h and $h''(Z)$ is an interval overestimating the range of h'' which we calculated through natural interval extensions in this example.

The McCormick relaxations of f_9 on Z (using the supplementary material of [42]) are given as

$$\begin{aligned} f_9^{cv}(z) &= f_9(-0.5) + \frac{f_9(-0.5) - f_9(1)}{-0.5 - 1}(z + 0.5) \\ f_9^{cc}(z) &= \log(z+1) - z^2. \end{aligned}$$

Interval extensions obtained by the Taylor form provide $I_{9,T} \approx [-1.751, 0.385]$ while the exact range is given as $I_{9,e} \approx [-0.943, 0.177]$. The convex and concave subgradients of f_9^{cv} and f_9^{cc}, respectively, at the middle point 0.25 of Z are $s_9^{cv}(0.25) = \frac{f_9^{cv}(-0.5) - f_9^{cv}(1)}{-0.5-1}$ and $s_9^{cc}(0.25) = 0.3$, respectively. We construct the affine functions

$$\begin{aligned} f_9^{cv,sub}(0.25, z) &= f_9^{cv}(0.25) + s_9^{cv}(0.25)(z - 0.25) \text{ and} \\ f_9^{cc,sub}(0.25, z) &= f_9^{cc}(0.25) + s_9^{cc}(0.25)(z - 0.25). \end{aligned}$$

Subsequently, we compute the respective minimum and maximum of the affine functions

$$\begin{aligned} \min_{z \in [-0.5, 1]} f_9^{cv,sub}(0.25, z) &= f_9^{cv,sub}(0.25, -0.5) \approx -0.943 \text{ and} \\ \max_{z \in [-0.5, 1]} f_9^{cc,sub}(0.25, z) &= f_9^{cc,sub}(0.25, 1) \approx 0.385. \end{aligned} \tag{3}$$

With (3) we can improve the interval bounds obtained by the Taylor form from $I_{9,T} = [-1.751, 0.385]$ to $I_{9,alg} = [-0.943, 0.385]$ for factor f_9. Factor f_9 together with its convex and concave relaxations (constructed with Taylor form bounds), Taylor form bounds $f_{9,T}^L, f_{9,T}^U$ and the affine functions can be seen in Fig. 4a. Similar to Example 2, this procedure can be rerun but to keep this example simple, we do only one iteration.

Next, we calculate improved range bound for factor f_{10}. The convex and concave McCormick relaxations of f_{10} on Z (using the supplementary material

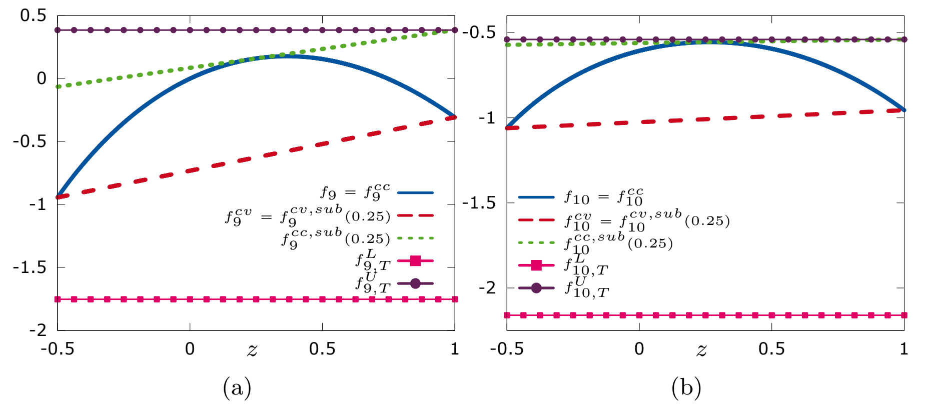


Fig. 4: Example 3. **(a)** Factor $f_9(z) = \log(z+1) - z^2$ with its convex and concave McCormick relaxations, Taylor form interval extension estimators $f_{9,T}^L, f_{9,T}^U$ for the range of f_9 on $Z = [-0.5, 1]$ and affine under- and overestimators constructed with the use of subgradients at the middle point 0.25. The affine underestimator equals the convex relaxation of f_9.
(b) Factor $f_{10}(z) = \log(z-1) - \exp(z-0.5)$ with its convex and concave McCormick relaxations, Taylor form interval extension estimators $f_{10,T}^L, f_{10,T}^U$ for the range of f_{10} on $Z = [-0.5, 1]$ and affine under- and overestimators constructed with the use of subgradients at the middle point 0.25. The affine underestimator equals the convex relaxation of f_{10}.

of [42]) are

$$\begin{aligned} f_{10}^{cv}(z) &= f_{10}(-0.5) + \frac{f_{10}(-0.5) - f_{10}(1)}{-0.5-1}(z+0.5) \\ f_{10}^{cc}(z) &= \log(z+1) - \exp(z-0.5). \end{aligned}$$

The Taylor form interval extensions provide $I_{10,T} \approx [-2.16, -0.539]$, while the exact range is given as $I_{10,e} \approx [-1.061, -0.555]$. The convex and concave subgradients of f_{10}^{cv} and f_{10}^{cc}, respectively, at the middle point 0.25 of Z are $s_{10}^{cv}(0.25) = \frac{f_{10}(-0.5)-f_{10}(1)}{-0.5-1}$ and $s_{10}^{cc}(0.25) = 0.8 - \exp(-0.25)$. We construct the corresponding affine functions

$$\begin{aligned} f_{10}^{cv,sub}(0.25, z) &= f_{10}^{cv}(0.25) + s_{10}^{cv}(0.25)(z - 0.25) \text{ and} \\ f_{10}^{cc,sub}(0.25, z) &= f_{10}^{cc}(0.25) + s_{10}^{cc}(0.25)(z - 0.25) \end{aligned}$$

and compute the respective minimum and maximum

$$\begin{aligned} \min_{z\in[-0.5,1]} f_{10}^{cv,sub}(0.25, z) &= f_{10}^{cv,sub}(0.25, -0.5) \approx -1.061 \text{ and} \\ \max_{z\in[-0.5,1]} f_{10}^{cc,sub}(0.25, z) &= f_{10}^{cc,sub}(0.25, 1) \approx -0.539. \end{aligned} \tag{4}$$

With (4) we can improve the interval bounds obtained by the Taylor form $I_{10,T} \approx [-2.16, -0.539]$ to $I_{10,alg} = [-1.061, -0.539]$ for factor f_{10}. Factor f_{10} together with its convex and concave McCormick relaxations (constructed with Taylor form bounds), Taylor form bounds $f_{10,T}^L, f_{10,T}^U$ and the affine functions

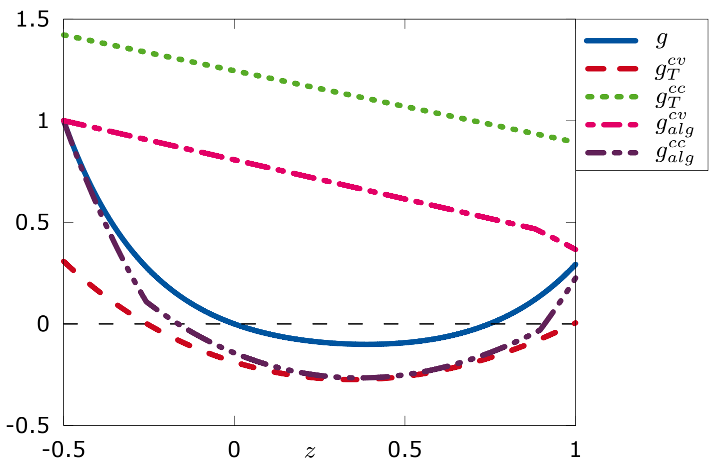


Fig. 5: Example 3. Function $g(z) = (\log(z+1) - z^2)(\log(z+1) - \exp(z-0.5))$ on $Z = [-0.5, 1]$ together with its convex and concave McCormick relaxations g_T^{cv}, g_T^{cc} constructed with Taylor form interval extensions and the convex and concave McCormick relaxation $g_{alg}^{cv}, g_{alg}^{cc}$ constructed using the range bounds computed via Algorithm 1 with only one iteration at each factor.

can be seen in Fig. 4b. Just as before, these steps can be recalculated but to keep this example simple, we do only one iteration.

Now, we are able to construct the envelope of $f_{11} = f_9 f_{10}$ on $I_{9,alg} \times I_{10,alg}$ and subsequently the McCormick relaxations of g on Z. Figure 5 shows function g together with its convex and concave McCormick relaxations constructed with the intervals $I_{9,T}, I_{10,T}$ obtained with the Taylor form interval extensions denoted as g_T^{cv}, g_T^{cc} and two improved McCormick relaxations constructed with the intervals $I_{9,alg}, I_{10,alg}$ denoted as $g_{alg}^{cv}, g_{alg}^{cc}$. We see that even for the more sophisticated interval extensions, the proposed method is able to significantly improve the final resulting McCormick relaxations.

Example 3 shows that the algorithm is able to improve the McCormick relaxations of a given function g even if more advanced interval arithmetic are used for the computation of range bounds of each factor. Obviously, it holds that the weaker the underlying estimated bounds for each factor, the larger is the potential of the presented heuristic.

3.2.3 Bivariate example

Since univariate functions can often be handled more efficiently, we present a bivariate example to better illustrate the merit of the method.

Example 4 Consider the two dimensional function $g : [-0.25, 1] \times [0.5, 1.5] \to \mathbb{R}, (x, y) \mapsto \left(x^2 \cdot (\log(y) + \exp(-y)) - x \cdot (\log(y) + \exp(-y))\right)^3$. First, examine the factor $f_6(y) = \log(y) + \exp(-y)$ with its convex and concave McCormick

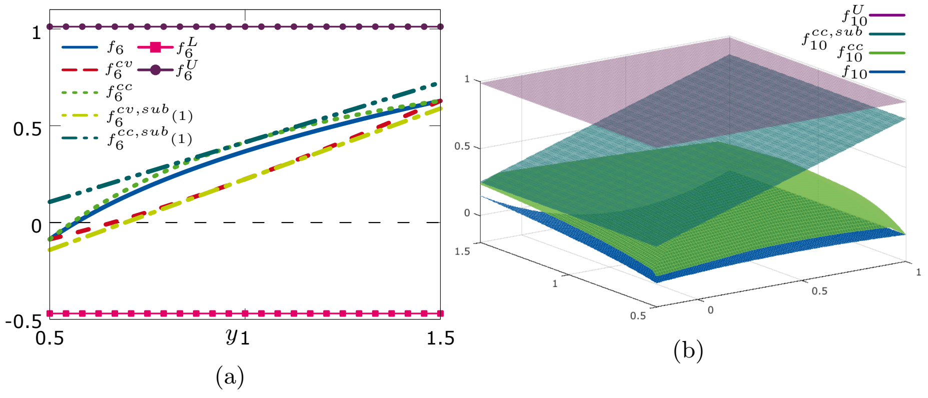


Fig. 6: Example 4. **(a)** Factor $f_6(y) = \log(y) + \exp(-y)$ with its convex and concave McCormick relaxations, natural interval extension estimators f_6^L, f_6^U for the range of f_6 on $Z = [0.5, 1.5]$ and affine under- and overestimators constructed with the use of subgradients at the middle point 1.
(b) Factor $f_{10}(x,y) = x^2 \cdot (\log(y) + \exp(-y)) - x \cdot (\log(y) + \exp(-y))$ with its concave McCormick relaxation f_{10}^{cc}, concave subgradient $f_{10}^{cc,sub}$ at the middle point $(0.375, 1)$ and the upper bound obtained by natural interval extensions using the improved bounds of factor f_6.

relaxations given as

$$f_6^{cv}(y) = \log(0.5) + \frac{\log(1.5) - \log(0.5)}{1.5 - 0.5}(y - 0.5) + \exp(-y)$$
$$f_6^{cc}(y) = \log(y) + \exp(-0.5) + \frac{\exp(-1.5) - \exp(-0.5)}{1.5 - 0.5}(y - 0.5)$$

and the (non-exact) natural interval extensions $I_{6,nat} \approx [-0.47, 1.011]$. The convex and concave subgradient of f_6^{cv} and f_6^{cc}, respectively, at the middle point 1 of Y are $s_6^{cv}(0.5) \approx 0.7307$ and $s_6^{cc} \approx 0.6165$, respectively. We construct the affine functions

$$f_6^{cv,sub}(1,y) = f_6^{cv}(1) + s_6^{cv}(1)(y - 0.5) \text{ and}$$
$$f_6^{cc,sub}(1,y) = f_6^{cc}(1) + s_6^{cc}(1)(y - 0.5).$$

Now, we evaluate the affine functions at their respective minimum and maximum and obtain

$$\begin{aligned}\min_{y\in[0.5,1.5]} f_6^{cv,sub}(1,y) &= f_6^{cv,sub}(1,0.5) \approx -0.1413 \text{ and}\\ \max_{y\in[0.5,1.5]} f_6^{cc,sub}(1,y) &= f_6^{cc,sub}(1,1.5) \approx 0.7231\end{aligned} \tag{5}$$

With (5) we can improve the interval bounds $I_{6,nat}$ from $[-0.47, 1.011]$ to $[-0.1413, 0.7231]$ for factor f_6. Factor f_6 together with its convex and concave McCormick relaxations (constructed with natural interval bounds), natural interval bounds f_6^L, f_6^U and the affine functions are shown in Fig. 6a.

The next factor where an improvement of the interval bounds occurs is $f_{10}(x,y) = x^2 \cdot (\log(y) + \exp(-y)) - x \cdot (\log(y) + \exp(-y))$. The concave sub-

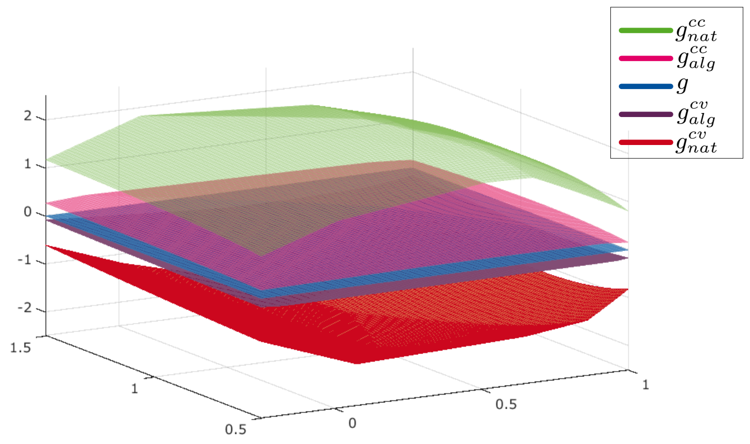


Fig. 7: Example 4. Function $g(x, y) = (x^2 \cdot (\log(y) + \exp(-y)) - x \cdot (\log(y) + \exp(-y)))^3$ on $X \times Y = [-0.25, 1] \times [0.5, 1.5]$ together with its convex and concave McCormick relaxations $g^{cv}_{nat}, g^{cc}_{nat}$ constructed with natural interval extensions and the convex and concave McCormick relaxation $g^{cv}_{alg}, g^{cc}_{alg}$ constructed using the range bounds computed via Algorithm 1 with only one iteration at each factor.

gradient of the concave McCormick relaxation f^{cc}_{10} is given by

$$f^{cc,sub}_{10}((0.375, 1)^T, (x, y)^T) = 0.5452 + 0.5107 \cdot (x - 0.375).$$

The concave relaxation of f_{10} was calculated using formulas (5) and (6) in [31] and is not listed herein due to the very large formulas. Natural interval extensions provide $I_{10,nat} \approx [-0.8644, 0.9039]$, where the improved bounds for factor f_6 were used for the computation. The maximum of $f^{cc,sub}_{10}((0.375, 1)^T, (x, y)^T)$ is given as

$$\begin{aligned} \max_{x \in [-0.25,1], y \in [0.5,1.5]} & f^{cc,sub}_{10}((0.375, 1)^T, (x, y)^T) \\ &= f^{cc,sub}_{10}((0.375, 1)^T, (1, y)^T) \approx 0.8644. \end{aligned} \tag{6}$$

With (6), we are able to improve $I_{10,nat}$ from $[-0.8644, 0.9039]$ to $[-0.8644, 0.8644]$, see Fig. 6b.

Last, we construct the envelopes of $f_{11} = f^3_{10}$ on the improved interval $I_{10,alg}$ and the convex and concave McCormick relaxation of g. Figure 7 shows the two dimensional function g together with its convex and concave McCormick relaxation constructed with the use of simple intervals denoted as $g^{cv}_{nat}, g^{cc}_{nat}$ and the improved McCormick relaxations constructed with the use of the presented method denoted as $g^{cv}_{alg}, g^{cc}_{alg}$.

3.3 Formal statement of the algorithm

For a given factorable function $g : Z \to \mathbb{R}, Z \in \mathbb{IR}^n$, we traverse the corresponding DAG of g starting at the independent variables $z_i, i \in \{1, \dots, n\}$ and working through all factors up to the root given as g (reverse-level-order in Graph Theory terminology). In each factor $f_j, j \in \{1, \dots, |\mathbb{F}|\}$, we execute Algorithm 1 in order to obtain bounds $I_{j,alg} = [f^L_{j,alg}, f^U_{j,alg}]$ on the range of f_j and save these.

We use $I_{j,alg}$ then directly when computing $f_k, k > j$. Note that the computed range bounds for every f_j are valid on whole Z. Thus, after the DAG of g has been completely traversed, we can calculate McCormick relaxations of g and its subgradients at any point $\bar{\mathbf{z}} \in Z$ with the use of the range bounds $I_{j,alg}$ for each factor $f_j, j \in \{1, \ldots, |\mathbb{F}|\}$ instead of using natural interval extensions for the range bounds estimation.

3.4 Algorithm discussion

We now discuss Algorithm 1. First, we check if the factor f_j we currently consider, is a constant function by simply comparing the lower and upper bound (line 4), since if it is the case, we cannot improve any bounds on its range and thus, the algorithm is unnecessary. Obviously this check is only sufficient and not necessary, since it is possible to have different bounds for a constant function due to overestimation. In the `for` loop (line 7), we solve $\min\limits_{\mathbf{z}\in Z} f_j^{cv,sub}(\bar{\mathbf{z}}, \mathbf{z})$ and $\max\limits_{\mathbf{z}\in Z} f_j^{cc,sub}(\bar{\mathbf{z}}, \mathbf{z})$ by simple subgradient comparisons as both problems are box-constrained and linear. The correct corner $\mathbf{z}^c$ of Z is determined by examination of the sign of the subgradient in the particular variable (lines 8 and 13). Then, we check if we can improve the bounds of factor f_j (lines 19 and 20). We update the range bounds I_j of factor f_j (line 21) such that they can be directly used for the computation of relaxations, subgradients and bounds for factors $f_k, k > j$. Then, we save the improved bounds (line 22). If needed, we compute a new point $\bar{\mathbf{z}}_{new}$ (line 24). There are many ways to compute a new point. In our computational studies in Section 4, we use the simple bisection in order to obtain a new $\bar{\mathbf{z}}_{new}$, i.e., the next point is given by the middle point of the interval $[\bar{\mathbf{z}}, \mathbf{z}^c]$, where $\mathbf{z}^c$ is computed before in lines 8 and 13 (not explicitly shown in Algorithm 1). Note that the simple bisection method converges to $\min\limits_{\mathbf{z}\in Z} f_j^{cv}(\mathbf{z})$ for $N \to \infty$. A more sophisticated method for the computation of $\bar{\mathbf{z}}_{new}$ may provide better results and represents potential future work. Note that it may also make sense to compute two new separate points in order to independently improve the upper and lower bound within the algorithm but the numerical results for one common new point are very weak (Table 4, Figure 11) and thus, we omit this idea.

We are also interested in the computational complexity of the presented method. The computation of the possibly improved bounds $I_{j,alg}{=}[f^L_{j,alg}, f^U_{j,alg}]$ (lines 5 - 22) is linear in the dimension of the optimization variables $\mathbf{z}$, i.e., we have a complexity of $\mathcal{O}(n)$ for the computation of $I_{j,alg}$. This is comparable to the computation of McCormick relaxations and propagation of subgradients which have to be computed for each dimension in each factor. If only one iteration is allowed ($N = 1$), we can directly use the propagated relaxation and subgradient values at the desired point $\bar{\mathbf{z}}$ and are not forced to re-evaluate McCormick relaxations, subgradients and interval bounds for all the factors on which f_j depends for subsequent points. Therefore, if a function g consists of $|\mathbb{F}|$ factors and we allow only one iteration of the algorithm, the computational complexity amounts to $\mathcal{O}(|\mathbb{F}| \cdot n)$. Regarding the computational time for $N = 1$, we can expect that even in cases where the algorithm does not yield any improvement, the additional computational effort is negligible. This observation is also confirmed in the numerical studies in Section 4. In contrast, if we allow for more than one iteration ($N > 1$) within a factor f_j, we have to propagate all

Algorithm 1: Method for obtaining tighter interval range bounds with the use of propagated subgradients for one factor f_j of g.

```
1  Given a DAG representation $G = (\mathbb{F}, E)$ of a factorable function
   $g : Z \to \mathbb{R},\ Z \in \mathbb{IR}^n$.
   $\mathbf{z}^L, \mathbf{z}^U$ - lower and upper, finite, bounds for all independent variables
   $z_i,\ i \in \{1, \dots, n\}$
   $\bar{\mathbf{z}}$ - initial point for computation of relaxations and subgradients
   $N$ - maximal number of iterations for given factor
   Initialize factor $f_j \in \mathbb{F}$ obtained by traversing $G$ in reversed-level-order to
   obtain $f_j^{cv}(\bar{\mathbf{z}}), f_j^{cc}(\bar{\mathbf{z}}), \mathbf{s}_j^{cv}(\bar{\mathbf{z}}), \mathbf{s}_j^{cc}(\bar{\mathbf{z}}), I_j = [f_j^L, f_j^U]$.
2  k=1;
3  while $k \leqslant N$ do
4      if $f_j^L < f_j^U$ then
5          $\mathrm{t}^{cv} = f_j^{cv}(\bar{\mathbf{z}})$;
6          $\mathrm{t}^{cc} = f_j^{cc}(\bar{\mathbf{z}})$;
7          for $i$=1,...,$n$ do
8              if $s_i^{cv}(\bar{\mathbf{z}}) \geqslant 0$ then
9                  $\mathrm{t}^{cv}$=$\mathrm{t}^{cv}$+$s_i^{cv}(\bar{\mathbf{z}})(z_i^L - \bar{z}_i)$;
10             else
11                 $\mathrm{t}^{cv}$=$\mathrm{t}^{cv}$+$s_i^{cv}(\bar{\mathbf{z}})(z_i^U - \bar{z}_i)$;
12             end
13             if $s_i^{cc}(\bar{\mathbf{z}}) \geqslant 0$ then
14                 $\mathrm{t}^{cc}$=$\mathrm{t}^{cc}$+$s_i^{cc}(\bar{\mathbf{z}})(z_i^U - \bar{z}_i)$;
15             else
16                 $\mathrm{t}^{cc}$=$\mathrm{t}^{cc}$+$s_i^{cc}(\bar{\mathbf{z}})(z_i^L - \bar{z}_i)$;
17             end
18         end
19         $f_{j,alg}^L = \max\{\mathrm{t}^{cv}, f_j^L\}$;
20         $f_{j,alg}^U = \min\{\mathrm{t}^{cc}, f_j^U\}$;
21         $I_j = [f_{j,alg}^L, f_{j,alg}^U]$;
22         Save $I_{j,alg} = I_j$;
23         if $k + 1 \leqslant N$ then
24             Recompute $\bar{\mathbf{z}}$;
25         end
26     end
27     k=k+1;
28 end
```

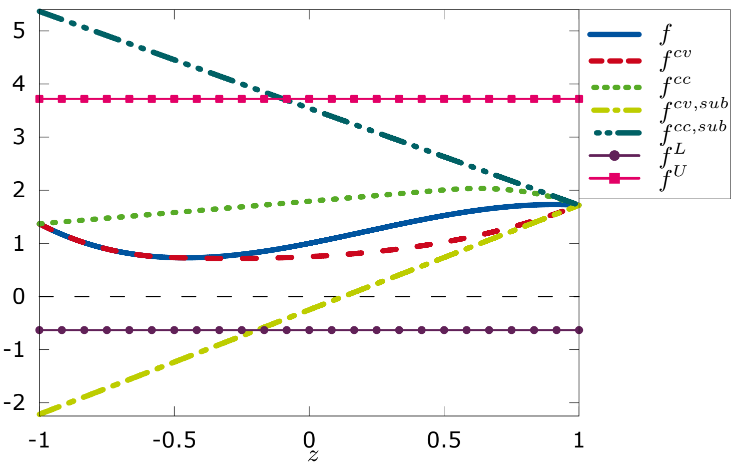


Fig. 8: Example 5. Factor $f(z) = \exp(z) - z^3$ on $Z = [-1, 1]$. The heuristic does not provide any improvement if the initial point is chosen as $\bar{z} = 1$ and only 1 iteration ($N = 1$) is allowed.

required information through all previous factors at the new point. Indeed, if a factor f_j depends on $\mathcal{J}$ other factors and we allow N iterations, we need to do $\mathcal{J}^N$ computations, which is large for more complicated functions ($\mathcal{J} \gg 1$) and $N > 1$. Moreover, let us assume that each factor depends on all the previous factors, then the complexity for N computations of f_j equals $\sum\limits_{k=1}^{j} k^N$, which is a polynomial complexity but still extremely large if a function consists of many factors and $N > 1$. The impact of the number of factors for $N > 1$ matches the numerical results presented in Section 4. To avoid a large number of re-computations, we could heuristically decide whether it is worth to traverse the DAG again, e.g., by the value of j or if the difference between the McCormick relaxation f_j^{cv} and the natural interval bounds f_j^L is very large.

3.5 Algorithm properties and limitations

The algorithm cannot deteriorate the bounds, since new bounds $I_{j,alg}$ are given by the maximum and minimum of the originally computed bounds f_j^L, f_j^U and t^{cv}, t^{cc} (lines 19 and 20). The best bounds obtained by Algorithm 1 cannot be better than the minimum and maximum of the convex and concave McCormick relaxations of a factor f_j, i.e., it holds that

$$f_{j,alg}^L \leqslant \min_{\mathbf{z}\in Z} f^{cv}(\mathbf{z}) \quad \text{and} \quad f_{j,alg}^U \geqslant \max_{\mathbf{z}\in Z} f^{cc}(\mathbf{z}).$$

Algorithm 1 is able to improve the range bounds of a factor, see Example 2 but is not guaranteed to improve the bounds of a factor f_j, e.g., if the underlying interval extensions for the bounds f_j^L, f_j^U are already exact or if the point $\bar{\mathbf{z}}$ is chosen badly as we show in the next example making the method a heuristic.
Example 5 Consider $f(z) = \exp(z) - z^3$ on $Z = [-1, 1]$. Let us apply Algorithm 1 to f at point $\bar{z} = 1$. The heuristic does not improve the interval bounds of f, see Fig. 8.

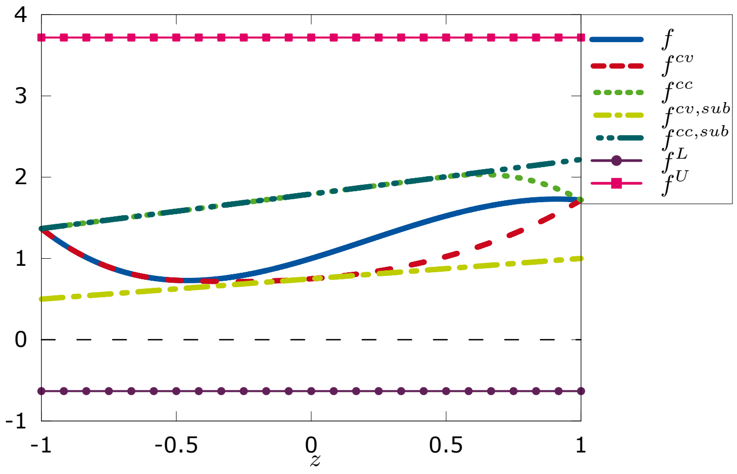


Fig. 9: Factor $f(z) = \exp(z) - z^3$ on $Z = [-1, 1]$. The heuristic provides clear improvement if an additional iteration is allowed or if the initial point is directly set to $\bar{z} = 0$ in Example 5.

Example 5 shows that the outcome of the method depends on the chosen initial point $\bar{\mathbf{z}}$ and also on the maximum number of iterations. Choosing a corner point $\mathbf{z}^c \in Z$ as initial point for Algorithm 1 is only a good choice if we have some monotonicity information of the convex relaxation. In general, choosing the initial point for the algorithm from the interior of Z seems more intuitive and promising due to the positive curvature of convex underestimators. Note that the bounds of f in Example 5 are improved if additional iterations of the heuristic are performed. If we allow for an additional iteration in Example 5, we obtain the middle point $\bar{\mathbf{z}}_{new} = 0$ by applying simple bisection of Z for the re-computation of $\bar{\mathbf{z}}$ and the heuristic indeed does improve the range bounds of f, which can be seen in Fig. 9.

3.6 Algorithm alternatives and initial point determination

Instead of solving $\min\limits_{\mathbf{z}\in Z} f_j^{cv,sub}(\bar{\mathbf{z}}, \mathbf{z})$ and $\max\limits_{\mathbf{z}\in Z} f_j^{cc,sub}(\bar{\mathbf{z}}, \mathbf{z})$ for only one point in each factor as described in Section 3.4, we could consider multiple points simultaneously in each factor f_j. For each point, we compute the subgradient and use it to construct linearizations in each factor f_j resulting in a linear program (LP) for the lower and the upper interval bound each. The resulting LPs would be of the form:

$$\begin{aligned} &\min_{\mathbf{z}\in Z} \;\eta && \max_{\mathbf{z}\in Z} \;\mu \\ &\text{s.t. } f_j^{cv,sub}(\bar{\mathbf{z}}_k, \mathbf{z}) \leqslant \eta \;\forall\; k = 1 \dots P \qquad && \text{s.t. } f_j^{cc,sub}(\bar{\mathbf{z}}_k, \mathbf{z}) \geqslant \mu \;\forall\; k = 1 \dots P, \end{aligned}$$

where P is the number of points considered. Note that in general every factor (except for the original variables and trivial cases) has to be inspected, since natural interval extensions do not guarantee exact bounds as soon as a variable occurs more than once or a multivariate operation, e.g., subtraction, is performed. This would result in the solution of twice as many LPs as there are factors for the determination of improved intervals of the whole function g for only one node in the B&B tree. Additionally, in order to achieve a worthy improvement through the solution of the high number of linear programs, the multiple points considered simultaneously in each factor f_j have to be chosen properly. In AVM the choice of linearization points for each factor is very simple since each factor has its own auxiliary variables and corresponding bounds. In the McCormick propagation technique, the choice of linearization points is far from simple. Thus, we present an alternative to an optimal choice of linearization points where we determine $(P-1)$ random points for linearization plus the middle point. A comparison of this approach with Algorithm 1 is presented in the next section.

4 Numerical results

In order to test the presented algorithm, we use the in-house deterministic global optimization solver MAiNGO [8]. In particular we use CPLEX v12.8 [19] for linear optimization, the IPOPT solver v3.11.0 [49] for local nonlinear optimization, the FADBAD++ package for automatic differentiation [2] and the MC++ package v2.0 [11] for McCormick relaxations. All calculations are conducted on an Intel® Core™i3-3240 with 3.4GHz and 8 GB RAM on Windows 7. We solve 15 small problems of varying sizes with up to 14 variables chosen arbitrarily from the COCONUT library [43]; 3 case studies of a combined-cycle power plant presented in Section 5 of [7] where we minimize the levelized cost of electricity in the first 2 case studies and maximize the net power output in the third case study and the 2 case studies for parameter estimation presented in Section 5 of [28]. The McCormick technique was already shown to be advantageous for the case studies from [7, 28] in the respective article, while the case studies from the COCONUT library were not solved with a McCormick based solver before. We solve the nonlinear optimization problems as explained in the following. The upper bounding problems are solved locally with IPOPT in order to obtain a valid upper bound. For the lower bound, we relax the problems using McCormick relaxations and then construct linearizations $g^{cv,sub}(\bar{\mathbf{z}}, \mathbf{z})$ at a single point $\bar{\mathbf{z}} \in Z$ with the use of subgradient propagation. The considered problems consist of constrained and box-constrained problems. In both cases we linearize the convex relaxation of the objective function and all constraints (excluding the variable bounds) at a predefined single point to construct a linear program, which is then solved with CPLEX. In the case of box-constraints only, we obtain an extremely simple linear program whose solution can be obtained by simple coefficient analysis. Still, even in this simple case we automatically call CPLEX for the solution. We always use only one linearization point, namely the middle point of the current node in the first and third numerical comparisons (Tables 2 and 4) and the current incumbent in the second comparison (Table 3). If any of the coordinates of the incumbent is not within a given node, we simply replace it by the corresponding middle point coordinate.

Additionally, in most problems, there are no bounds given for the optimization variables. Since McCormick relaxations need valid bounds in order to be constructed, we provide valid bounds containing the global minimum in all problems, see Table 1 in Appendix A for the number of variables, inequalities, equalities and bounds we used. If the bounds are provided in the given optimization problem, we mark it with the keyword *given*. Moreover, problem `ex6_2_10` consists of many occurrences of the convex function $f(x) = x \cdot \log(x)$. In this particular problem and in problem `ex8_5_6` we use the `xlog` function, implemented in MC++(v2.0) [11] instead of using the binary product of x and $\log(x)$. We also use the envelope of the logarithmic mean temperature difference function, presented in [32] for the 3 case studies from [7] explaining the improved computational times for $N = 0$, i.e., without the use of Algorithm 1, in this article in comparison to [7]. For the second parameter estimation case study in [28], we reformulate the problem by simply factorizing the relevant variable in each Euler-method equation, i.e., we reformulated $x_{i,n+1} = x_{i,n} + \Delta t \cdot f(\mathbf{x}_n)$ to $x_{i,n+1} = x_{i,n} \cdot g(\mathbf{x}_n)$ with $i \in \{A, B, Z, Y, D\}$ in this particular problem. This results in a drastically faster convergence compared to the formulation in [28] for our approach. In the case of BARON no significant advantage is observed. Thus, after this reformulation the method discussed in [28] shows a drastic advantage over the state-of-the-art solver BARON as can be seen in the last row of Table 2 for **MC only** compared to BARON.

In the following, we discuss the numerical impact of the method within a B&B framework using McCormick relaxations. The presented algorithm is not applicable to the αBB method [1, 24], since the αBB method does not depend on tight bounds of intermediate factors of a function g but rather on bounds on the eigenvalues of g. Algorithm 1 is also not applicable (at least not directly) to the AVM, where an auxiliary variable is introduced for each factor, instead of propagating the obtained relaxations and bounds. Still, we present a comparison with the state-of-the-art deterministic global optimization solver BARON [47] which uses the AVM. It is worth mentioning that the advantages of the described B&B procedure in the sense of computational time compared to state-of-the-art deterministic global optimization solvers has already been shown in [7, 28] for numerical experiments `Case study I,II,III` [7], `heat` and `kinetic` [28].

In all numerical experiments we set the absolute and relative optimality tolerances to $\epsilon = 10^{-4}$ and absolute and relative feasibility tolerances to $\epsilon = 10^{-6}$. First, we compare the impact of the algorithm with only one iteration. We allowed for a maximum of 3600 seconds. We consider the computational performance of B&B for four cases: with and without the heuristic, as well as with and without range reduction (RR). Range reduction is performed using *Optimization Based Bound Tightening* improved by the filtering bounds technique with factor 0.5 described in [16] and also employing bound tightening based on the dual multipliers returned by CPLEX [38]. In the figures the case of no RR and no heuristic is referred to as **(MC only)**, the case of no RR and heuristic used as **(MC heur)**, the case of RR used and no heuristic as **(MC RR)** and the case of using RR and the heuristic as **(MC heur RR)**.

Table 2 in Appendix A summarizes the results for only one allowed iteration within Algorithm 1. Figures 10a and 10c show performance plots for the heuristic applied at the midpoint. We observe that the algorithm has the potential to drastically decrease the number of iterations and the computational time needed. The effect is especially high in the case studies from [7] where the objective function and the constraints are composed of many factors and the propagated interval bounds of each factor lose on quality quickly.

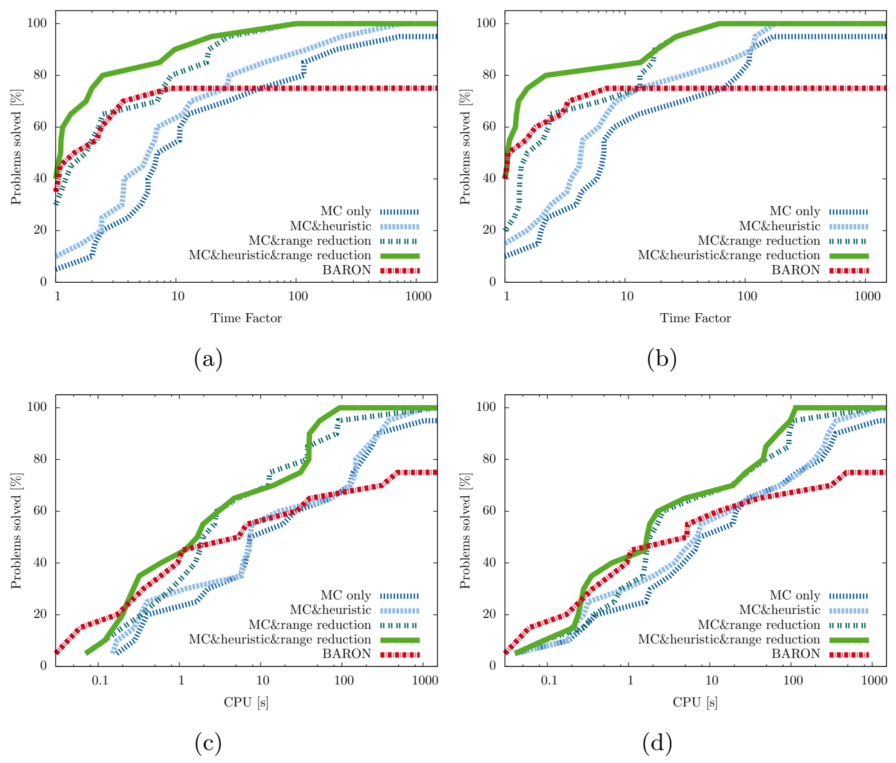


Fig. 10: Performance plots with time factors and CPU time needed in seconds. In all subfigures we are comparing the in-house solver MAiNGO [8] with and without additional range reduction and the presented method with the state-of-the-art solver BARON v17.10.16 with default settings. The x-axis is in logarithmic scale for a better comparison.
(a) and **(c)** Algorithm 1 applied at the midpoint for $N = 1$.
(b) and **(d)** Algorithm 1 applied at the incumbent for $N = 1$.

Moreover, in the cases where the heuristic did not improve the relaxations, the number of iterations remained the same and the computational time only increased, if at all, by a very marginal amount. This is explained by the fact that if only one iteration of the heuristic is allowed, we can directly integrate the heuristic into the computation of McCormick relaxations and the heuristic only has a constant computational complexity in each propagation step. It is worth noting that the cases where the heuristic did not provide improvement are relatively simple and solvable within a few seconds with additional range reduction. Additionally, we compare the results to the state-of-the-art solver BARON v17.10.16 [47], where we used default settings, see Fig. 10a. Such comparisons of BARON and a McCormick-based B&B solver were already made in [7, 28]. We also observe that the McCormick-based B&B solver outperforms BARON with and without additional range reduction and the presented heuristic for the hard case studies from [7, 28]. For most benchmark problems which are solved within few seconds, the state-of-the-art solver slightly outperforms the methods used herein.

Next, we compare the impact of the heuristic with only one iteration but

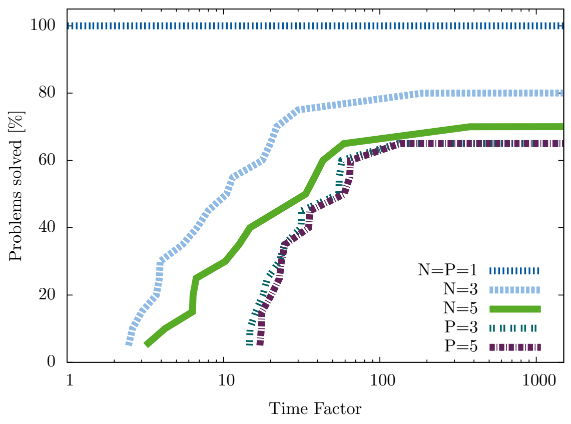


Fig. 11: Performance plot comparing the heuristic for $N = P = 1, N = 3$ and $N = 5$ applied at midpoint and solving LP with $P = 3$ and $P = 5$ linearization points in each factor without additional range reduction.

a different initial point. Figures 10b and 10d show performance plots for the heuristic applied at the incumbent. Again, we allowed for a maximum of 3600 seconds. This time, instead of the middle point of the node, we use the incumbent found by the local solver in the upper bounding procedure as the initial point *and* as the only linearization point in order to construct the linear lower bounding problem. If any coordinate of the incumbent is not within the current node, we simply use the appropriate coordinate of the middle point of this node instead. Table 3 in Appendix A summarizes the results for only 1 allowed iteration within Algorithm 1 with the current incumbent as initial point. Again, especially in the three case studies from [7], the heuristic improves the solution times drastically. We see again that the computational time only increased, if at all, by a very marginal amount. We observe that the choice of the initial point may have a significant impact on the advantage provided by the heuristic. Again, we compare the results to the state-of-the-art solver BARON v17.10.16 [47], where we used default settings.

Last, we compare the impact of the number of iterations within the heuristic and the solution of LPs in each factor. Table 4 summarizes the numerical results and Figure 11 shows the performance plot comparing the heuristic for $N = P = 1, N = 3$ and $N = 5$ applied at midpoint and solving LPs with multiple linearization points (Section 3.6) for $P = 3$ and $P = 5$ without any additional range reduction. We allow for a maximum of 3600 seconds. We see that the number of iterations in the B&B does not increase if we allow more iterations or more points for the LPs within the heuristic but the computational time needed explodes, especially for problems with many factors. This is the behavior that we already shortly discuss in Sections 3.4 and 3.6. Even if the number of factors within the optimization problems is not very high, the additional time needed for further iterations within each factor or the time needed for the solution of LPs adds up quickly and is clearly visible if many iterations are needed in order to solve a given optimization problem. Note that in this work we used a very simple way to compute additional iterations or to choose the linearization points for LPs, so an improved method for computing further iterations could improve the computational times.

5 Conclusion

We present a new heuristic for tightening of the univariate McCormick relaxations [25, 26] and its extension to multivariate outer functions [48] of a factorable function g based on the idea of using tighter interval bounds for the range of each factor of g and obtaining these through subgradients, presented in Section 2.3 in [33] and Example 4.4 in [28]. The algorithm possibly improves the range bounds of the factors of g. It uses subgradient propagation for McCormick relaxations [28] in order to construct simple valid affine under- and overestimators of each factor. Then, the affine relaxations are solved with simple function evaluations resulting in improved range bounds for each factor. This results in tighter McCormick relaxations of the original function g.

Subsequently, we provide numerical results confirming the potential of the presented heuristic. We observe that allowing for only one iteration within the heuristic results in the best computational times. Although more iterations give potentially better bounds, this leads to recalculation of a possibly high number of factors of a factorable function g. Moreover, we see that selecting a good initial point may significantly improve the outcome of the algorithm, which remains a potential future work regarding the presented algorithm. Algorithm 1 is especially effective if the underlying interval bounds of the factors are very loose. This is often the case when the well-known dependency problem applies [29, 37]. A combination of the presented method with the reverse propagation of McCormick relaxations presented in [51] appears to be promising, since it works with range bounds of g and could result in an even greater improvement of McCormick relaxations overall, representing a further potential future development for the McCormick technique. One could also think of a combination of the auxiliary variable method [46] and the McCormick technique to isolate problematic factors by introduction of auxiliary variables and then directly applying the presented heuristic to these.

Acknowledgements We would like to thank Dr. Benoît Chachuat for providing MC++ v2.0 and Dominik Bongartz for providing numerical case studies. We appreciate the thorough review and helpful comments provided by the anonymous reviewers and editors which resulted in an improved manuscript. This project has received funding from the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) *Improved McCormick Relaxations for the efficient Global Optimization in the Space of Degrees of Freedom* MA 1188/34-1.

A Appendix

Table 1: Table summarizing the problems for the numerical studies for different numbers of iterations within the heuristic. #var represents the number of variables, #ineq stands for the number of inequalities and #eq for the number of equalities. The domain denotes the domain, we used for the variables, if it was not already *given*. The first 15 problems can be found in the COCONUT benchmark library [43].

Name	#var	#ineq	#eq	domain
alkyl	14	0	7	*given*
bard	3	0	0	$[0.001, 10]^3$
eg1	3	0	0	$[-10, 10]\times[-1, 1]\times[1, 2]$
ex3_1_1	8	6	0	*given*
ex6_2_10	6	0	3	*given*
ex6_2_11	3	0	1	*given*
ex6_2_14	4	0	2	*given*
ex8_1_3	2	0	0	$[-2, 2]^2$
ex8_5_6	6	0	4	$[10^{-6}, 10]^3\times[0.2, 10]\times[10^{-6}, 10]\times[10^{-6}, 0.19]$
growthls	3	0	0	$[1, 2]\times[0, 1]\times[0, 1]$
himmelbf	4	0	0	$[0, 10]\times[0, 10^3]\times[0, 10^5]\times[-100, 0]$
meanvar	8	0	2	*given*
mh4wd	5	0	3	$[-1000, 1000]^5$
process	10	0	7	*given*
rosenmmx	5	4	0	$[-10, 10]^4\times[-100, 100]$
CS I[7]	2	0	9	*given*
CS II[7]	5	1	12	*given*
CS III[7]	8	14	1	*given*
heat [28]	1	0	0	*given*
kinetic [28]	3	0	0	*given*

Table 2: Table summarizing the numerical results. We allowed only 1 iteration within the heuristic and used the middle point of each node as its initial point. If the time limit of 3600s has been reached, we provide the reached convergence ratio in percent. The problem `eg1` contains trigonometric functions and thus could not be solved with BARON.

$N = 1$	MC only		MC heur		MC RR		MC heur RR		BARON	
Name	#iter	CPU[s]	#iter	CPU[s]	#iter	CPU[s]	#iter	CPU[s]	#iter	CPU[s]
alkyl	60209	26.72	12645	6.02	527	1.79	73	0.23	3	0.25
bard	33799	6.98	33799	7.32	1705	1.18	1705	1.13	1	0.06
eg1	1039	0.17	1039	0.17	115	0.07	115	0.07	-	-
ex3_1_1	19239	5.55	19239	5.78	1177	1.95	1177	1.95	1225	1.09
ex6_2_10	529289	181.16	529289	203.22	96461	85.5	96461	93.97	69427	311
ex6_2_11	320759	73.96	315363	74.6	40101	30.96	37757	27.56	5747	12.6
ex6_2_14	32537	7.44	32169	7.86	1089	0.79	775	0.59	195	0.59
ex8_1_3	9881	1.63	993	0.15	6573	2.69	459	0.2	265	0.36
ex8_5_6	56699	18.82	48295	16.77	4265	5.21	3735	4.71	3212	6.71
growthls	4385257	1019.18	4385257	1065.11	57853	38.04	57853	39.46	561	40.14
himmelbf	1482749	269.5	1482749	291	54843	37.34	54843	39.21	2541	5.34
meanvar	17381	6.48	17381	7.23	1993	2.9	1993	3.02	0	0.03
mh4wd	550483	123.83	550483	128.03	1411	1.56	1411	1.66	9	0.17
process	5679	2.24	2487	1.17	405	0.79	111	0.32	699	0.95
rosenmmx	1401	0.4	1401	0.4	79	0.12	79	0.12	0	0.06
CS I [7]	625	0.4	467	0.34	71	0.2	57	0.2	201627	95.6%
CS II [7]	348991	233.53	202827	148.15	69405	91.16	27299	39.88	129200	90.88%
CS III [7]	3735040	91.93%	482761	373.63	725547	1459.08	26089	53.52	23629	484.12
heat [28]	129	0.28	121	0.28	107	0.34	103	0.32	35716	33.19%
kinetic [28]	29735	138.45	29663	147.93	1625	12.94	1569	14.32	1	93.80%

Table 3: Table summarizing the numerical results. We allowed only 1 iteration within the heuristic and used the incumbent as its initial point. If the incumbent was not within the node bounds, we used the middle point of the node instead. If the time limit of 3600s has been reached, we provide the reached convergence ratio in percent.

$N = 1$	MC only		MC heur		MC RR		MC heur RR	
Name	#iter	CPU[s]	#iter	CPU[s]	#iter	CPU[s]	#iter	CPU[s]
alkyl	49237	21.07	4209	2.02	579	1.51	69	0.24
bard	31905	6.55	31905	6.95	2349	1.6	2349	1.6
eg1	1035	0.18	1035	0.18	151	0.09	151	0.09
ex3_1_1	16161	4.74	16161	4.85	975	1.68	975	1.68
ex6_2_10	653705	233.98	653705	253.09	125087	104.72	125087	114.7
ex6_2_11	322007	73.78	316605	75.95	40215	29.04	37737	27.61
ex6_2_14	32535	7.61	32143	7.81	1107	0.81	815	0.62
ex8_1_3	10021	1.62	1159	0.2	6605	2.66	521	0.26
ex8_5_6	58025	19.21	50231	17.37	5263	6.45	3929	4.89
growthls	4427869	1197.68	4427869	1235.34	72381	48.03	72381	49.1
himmelbf	1380431	353.2	1380431	360.2	138315	94.94	138315	97.98
meanvar	8269	3.24	8269	3.63	1099	1.82	1099	1.82
mh4wd	129789	28.98	129789	30.07	2263	2.26	2275	2.29
process	5011	1.87	2209	0.92	329	0.67	115	0.28
rosenmmx	1	0.04	1	0.04	1	0.04	1	0.04
CS I [7]	623	0.4	425	0.32	71	0.28	55	0.21
CS II [7]	463649	302.54	283119	193.89	69665	93.75	31897	45.27
CS III [7]	3870075	93.37%	374355	285.38	603865	1167.36	33357	68.45
heat [28]	131	0.28	123	0.28	109	0.37	105	0.35
kinetic [28]	31949	118.01	31753	129.63	2717	18.5	2591	19.34

Table 4: Table summarizing the numerical results for different numbers of iterations within the heuristic. We used the midpoint as initial point. N denotes the number of iterations within the heuristic. P denotes the number of linearization points in each factor for LP solution. #iter gives the number of iteration needed in the B&B algorithm. $|\mathbb{F}|$ denotes the number of factors in the given optimization problem.

	$N=1$		$N=3$		$N=5$		$P=3$		$P=5$		
Name	#iter	CPU[s]	#iter	CPU[s]	#iter	CPU[s]	#iter	CPU[s]	#iter	CPU[s]	$\|\mathbb{F}\|$
alkyl	12645	6.02	12595	32.96	12595	61.83	12519	115.05	12429	137.39	80
bard	33799	7.32	33799	160.29	33799	316.66	33799	400.48	33799	432.94	171
eg1	1039	0.17	1039	0.42	1039	0.54	1039	2.49	1039	2.99	15
ex3_1_1	19239	5.78	19239	22.46	19239	38.67	19239	103.3	19239	116.95	55
ex6_2_10	529289	203.22	195988	99.12%	96548	94.12%	122974	96.52%	107297	95.28%	215
ex6_2_11	315363	74.6	314823	1488.45	314767	2848.14	250684	99.91%	218675	99.83%	129
ex6_2_14	32169	7.86	32169	139.12	32169	263.79	32169	449.09	32167	504.16	101
ex8_1_3	993	0.15	885	0.59	851	0.95	889	3.68	795	3.52	51
ex8_5_6	48295	16.77	47341	133.42	47341	247.65	47719	525.19	47719	588.45	97
growthls	4385257	1065.11	738286	99.51%	421078	99.67%	276314	99.45%	247320	99.45%	122
himmelbf	1482749	291	1482749	3045.12	829975	97.42%	538681	85.53%	489624	81.3%	106
meanvar	17381	7.23	17381	216.31	17381	425.38	17383	397.61	17383	469.12	164
mh4wd	550483	128.03	550483	335.07	550483	540.84	550483	2054.69	550483	2237.09	37
process	2487	1.17	2475	4.35	2475	7.48	2477	17.31	2469	19.99	73
rosenmmx	1401	0.4	1401	2.71	1401	5.03	1401	12.65	1401	14.24	74
CS I [7]	467	0.34	467	3.9	467	7.58	463	7.75	463	8.4	252
CS II [7]	202827	148.15	161850	99.97%	85230	99.85%	133312	99.94%	118628	99.92%	333
CS III [7]	482761	373.63	165329	96.41%	83530	92.42%	119014	94.95%	101750	94.14%	264
heat [28]	121	0.28	121	52.72	121	105.28	121	36.58	117	37.68	1201
kinetic [28]	29663	147.93	29647	443.69	29647	660.35	3252	92.71%	2753	92.18%	4516

References

[1] I. P. Androulakis, C. D. Maranas, and C. A. Floudas. αBB: A global optimization method for general constrained nonconvex problems. *Journal of Global Optimization*, 7(4):337–363, 1995.

[2] C. Bendtsen and O. Staunin. *FADBAD++, a flexible C++ package for automatic differentiation. Version 2.1*, 2012. http://www.fadbad.com. Accessed 18 October 2016.

[3] D. P. Bertsekas. *Convex optimization algorithms*. Athena Scientific Belmont, 2015.

[4] D. P. Bertsekas, A. Nedic, A. E. Ozdaglar, et al. *Convex analysis and optimization*. Athena Scientific, 2003.

[5] A. Bompadre and A. Mitsos. Convergence rate of McCormick relaxations. *Journal of Global Optimization*, 52(1):1–28, 2012.

[6] A. Bompadre, A. Mitsos, and B. Chachuat. Convergence analysis of Taylor models and McCormick-Taylor models. *Journal of Global Optimization*, 57(1):75–114, 2013.

[7] D. Bongartz and A. Mitsos. Deterministic global optimization of process flowsheets in a reduced space using McCormick relaxations. *Journal of Global Optimization*, 2017.

[8] D. Bongartz, J. Najman, S. Sass, and A. Mitsos. MAiNGO - McCormick-based Algorithm for mixed-integer Nonlinear Global Optimization. Technical report, Aachener Verfahrenstechnik - Systemverfahrenstechnik, RWTH Aachen University, 2018.

[9] A. Brearley, G. Mitra, and H. P. Williams. Analysis of mathematical programming problems prior to applying the simplex algorithm. *Mathematical programming*, 8(1):54–83, 1975.

[10] A. L. Brearley, G. Mitra, and H. P. Williams. Analysis of mathematical programming problems prior to applying the simplex algorithm. *Mathematical Programming*, 8(1):54–83, 12 1975.

[11] B. Chachuat, B. Houska, R. Paulen, N. Peri'c, J. Rajyaguru, and M. E. Villanueva. Set-theoretic approaches in analysis, estimation and control of nonlinear systems. *IFAC-PapersOnLine*, 48(8):981 – 995, 2015. https://omega-icl.github.io/mcpp/(Accessed February 2017).

[12] J. L. D. Comba and J. Stolfi. Affine arithmetic and its applications to computer graphics. In *Proceedings of VI SIBGRAPI (Brazilian Symposium on Computer Graphics and Image Processing)*, pages 9–18. Citeseer, 1993.

[13] H. Cornelius and R. Lohner. Computing the range of values of real functions with accuracy higher than second order. *Computing*, 33(3-4):331–347, 1984.

[14] L. H. De Figueiredo and J. Stolfi. Affine arithmetic: concepts and applications. *Numerical Algorithms*, 37(1-4):147–158, 2004.

[15] K. Du and R. B. Kearfott. The cluster problem in multivariate global optimization. *Journal of Global Optimization*, 5(3):253–265, 1994.

[16] A. M. Gleixner, T. Berthold, B. Müller, and S. Weltge. Three enhancements for optimization-based bound tightening. *Journal of Global Optimization*, 67(4):731–757, 2017.

[17] A. S. E.-D. Hamed and G. P. McCormick. Calculation of bounds on variables satisfying nonlinear inequality constraints. *Journal of Global Optimization*, 3(1):25–47, 1993.

[18] P. Hansen, B. Jaumard, and S.-H. Lu. An analytical approach to global optimization. *Mathematical Programming*, 52(1):227–254, 1991.

[19] International Business Machines Corporation:. *IBM ILOG CPLEX v12.8. Armonk*, 2009.

[20] R. Kannan and P. I. Barton. The cluster problem in constrained global optimization. *Journal of Global Optimization*, 5 2017.

[21] R. Kannan and P. I. Barton. Convergence-order analysis of branch-and-bound algorithms for constrained problems. *Journal of Global Optimization*, 6 2017.

[22] B. Kearfott and K. Du. *The Cluster Problem in Global Optimization: the Univariate Case*, pages 117–127. Springer Vienna, Vienna, 1993.

[23] M. Locatelli and F. Schoen. *Global optimization: theory, algorithms, and applications*. SIAM, 2013.

[24] C. D. Maranas and C. A. Floudas. A global optimization approach for Lennard-Jones microclusters. *The Journal of Chemical Physics*, 97(10):7667–7678, 1992.

[25] G. P. McCormick. Computability of global solutions to factorable nonconvex programs: Part I-Convex underestimating problems. *Mathematical Programming*, 10:147–175, 1976.

[26] G. P. McCormick. *Nonlinear programming: Theory, Algorithms, and Applications.* Wiley, New York, 1983.

[27] R. Misener and C. Floudas. Antigone: Algorithms for continuous / integer global optimization of nonlinear equations. *Journal of Global Optimization*, 59(2-3):503–526, 2014.

[28] A. Mitsos, B. Chachuat, and P. I. Barton. McCormick-based relaxations of algorithms. *SIAM Journal on Optimization*, 20(2):573–601, 2009.

[29] R. E. Moore and F. Bierbaum. *Methods and applications of interval analysis (SIAM Studies in Applied and Numerical Mathematics).* Society for Industrial & Applied Math, 1979.

[30] D. Morrison, S. Jacobson, J. Sauppe, and E. Sewell. Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning. *Discrete Optimization*, 19:79–102, 2 2016.

[31] J. Najman, D. Bongartz, A. Tsoukalas, and A. Mitsos. Erratum to: Multivariate McCormick relaxations. *Journal of Global Optimization*, pages 1–7, 2016.

[32] J. Najman and A. Mitsos. Convergence Order of McCormick Relaxations of LMTD Function in Heat Exchanger Networks. In Z. Kravanja and M. Bogataj, editors, *26th European Symposium on Computer Aided Process Engineering*, volume 38 of *Computer Aided Chemical Engineering*, pages 1605 – 1610. Elsevier, 2016.

[33] J. Najman and A. Mitsos. On tightness and anchoring of McCormick and other relaxations. *Journal of Global Optimization*, Dec 2017.

[34] J. Ninin, F. Messine, and P. Hansen. A reliable affine relaxation method for global optimization. *4OR*, 13(3):247–277, 2015.

[35] Y. Puranik and N. V. Sahinidis. Bounds tightening based on optimality conditions for nonconvex box-constrained optimization. *Journal of Global Optimization*, 67(1-2):59–77, 2017.

[36] Y. Puranik and N. V. Sahinidis. Domain reduction techniques for global NLP and MINLP optimization. *Constraints*, pages 1–39, 2017.

[37] H. Ratschek and J. Rokne. *Computer methods for the range of functions.* E. Horwood ; Halsted Press Chichester : New York, 1984.

[38] H. S. Ryoo and N. V. Sahinidis. Global optimization of nonconvex NLPs and MINLPs with applications in process design. *Computers & Chemical Engineering*, 19(5):551–566, 1995.

[39] H. S. Ryoo and N. V. Sahinidis. A branch-and-reduce approach to global optimization. *Journal of Global Optimization*, 8(2):107–138, 1996.

[40] A. Sahlodin and B. Chachuat. Convex/concave relaxations of parametric odes using taylor models. *Computers & Chemical Engineering*, 35(5):844 – 857, 2011. Selected Papers from ESCAPE-20 (European Symposium of Computer Aided Process Engineering - 20), 6-9 June 2010, Ischia, Italy.

[41] H. Schichl and A. Neumaier. Interval analysis on directed acyclic graphs for global optimization. *Journal of Global Optimization*, 33(4):541–562, 2005.

[42] J. K. Scott, M. D. Stuber, and P. I. Barton. Generalized McCormick relaxations. *Journal of Global Optimization*, 51(4):569–606, 2011.

[43] O. Shcherbina, A. Neumaier, D. Sam-Haroud, X.-H. Vu, and T.-V. Nguyen. *Benchmarking Global Optimization and Constraint Satisfaction Codes*, pages 211–222. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003.

[44] J. P. Shectman and N. V. Sahinidis. A finite algorithm for global minimization of separable concave programs. *Journal of Global Optimization*, 12(1):1–36, 1 1998.

[45] E. M. Smith and C. C. Pantelides. Global optimisation of nonconvex MINLPs. *Computers & Chemical Engineering*, 21:791–796, 1997.

[46] M. Tawarmalani and N. V. Sahinidis. *Convexification and global optimization in continuous and mixed-integer nonlinear programming: theory, algorithms, software, and applications*, volume 65. Springer Science & Business Media, 2002.

[47] M. Tawarmalani and N. V. Sahinidis. A polyhedral branch-and-cut approach to global optimization. *Mathematical Programming*, 103(2):225–249, 2005.

[48] A. Tsoukalas and A. Mitsos. Multivariate McCormick Relaxations. *Journal of Global Optimization*, 59:633–662, 2014.

[49] A. Wächter and L. T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 3 2006.

[50] A. Wechsung, S. D. Schaber, and P. I. Barton. The cluster problem revisited. *Journal of Global Optimization*, 58(3):429–438, 2014.

[51] A. Wechsung, J. K. Scott, H. A. J. Watson, and P. I. Barton. Reverse propagation of McCormick relaxations. *Journal of Global Optimization*, 63(1):1–36, 9 2015.