Journal of Global Optimization
https://doi.org/10.1007/s10898-022-01265-6

Subdomain separability in global optimization

Jens Deussen[1] · **Uwe Naumann**[1]

Received: 26 October 2020 / Accepted: 13 December 2022

Abstract
We introduce a generalization of separability for global optimization, presented in the context of a simple branch and bound method. Our results apply to continuously differentiable objective functions implemented as computer programs. A significant search space reduction can be expected to yield an acceleration of any global optimization method. We show how to utilize interval derivatives calculated by adjoint algorithmic differentiation to examine the monotonicity of the objective with respect to so called structural separators and how to verify the latter automatically.

Keywords Global optimization · Algorithmic differentiation · Branch and bound · Interval adjoints · Search space reduction · Separable functions

Mathematics Subject Classification 65G30 · 90C26

1 Introduction

In contrast to local optimization methods, deterministic global optimization methods, e.g., interval-based branch and bound (b&b) algorithms [1], guarantee to find the global solution for a predefined tolerance for optimality in finite time [2, 3]. These methods are more expensive in terms of computational effort than their local counterparts.

An important property that should be exploited during optimization is the decomposition of the index set of variables of an optimization problem. We consider continuously differentiable objective functions $f : S \subset \mathbb{R}^n \to \mathbb{R}$ with an available evaluation formula $f(x)$, where $S \in \mathbb{I}^n$ and $\mathbb{I}$ stands for the set of all closed real intervals. We follow [4] and denote a decomposition of the index set of the variables $I = \{1, \ldots, n\}$ into p subsets as $I^{[j]} \subseteq I$ with $\bigcup_{j=1}^{p} I^{[j]} = I$. Each subset has $n^{[j]} = \left|I^{[j]}\right|$ elements and a corresponding part of the domain $S^{[j]} \in \mathbb{I}^{n^{[j]}}$. In this paper, we assume disjoint subsets, i.e., $I^{[j]} \cap I^{[k]} = \emptyset, \forall k, j \in \{1, \ldots, p\}, j \neq k$, such that there are no common variables.

✉ Jens Deussen
deussen@stce.rwth-aachen.de

[1] Informatik 12: Software and Tools for Computational Engineering, RWTH Aachen University, Aachen, Germany

A function f is called partially separable (also: decomposable or block-separable) if it is of the form

$$f(x) = \sum_{j=1}^{p} f^{[j]}\left(x^{[j]}\right),\ x^{[j]} \in S^{[j]}, \tag{1}$$

with a given partitioning of the variable indices into p disjoint subsets and sub-functions $f^{[j]} : S^{[j]} \subset \mathbb{R}^{n^{[j]}} \to \mathbb{R}$. A function is called (fully) separable [5, 6] if it is of the form (1) with $n^{[j]} = 1$, $j \in \{1, \ldots, p\}$.

Separability can be exploited for box constrained global optimization problems

$$y^* = \min_{x \in S} f(x), \tag{2}$$

with search space $S \in \mathbb{I}^n$ and partially separable objective function f as in (1). It is well known [7] that the global minimum and the corresponding minimizer of (2) can be obtained by decomposition into smaller optimization problems

$$y^* = \sum_{j=1}^{p} \min_{x^{[j]} \in S^{[j]}} f^{[j]}\left(x^{[j]}\right), \tag{3}$$

that can be solved in parallel. In the context of b&b algorithms it is easier to solve (3) than (2) due to the decomposition of the search space.

Separable functions have been extensively investigated in the context of optimization. In [8] a quasi-Newton method is introduced that exploits the structure of partially separable functions when computing secant updates for the Hessian matrix. A parallel b&b approach was used in [9] to find optima of non-convex problems with partially separable functions over a bounded polyhedral set. The automatic detection of partial separability as in (1) by algorithmic differentiation (AD) was presented in [10].

Another class of problems related to the present work is introduced in the context of incomplete global search in [11] and called *as easy to optimize as decomposable functions*. A function $f : S \subset \mathbb{R}^n \to \mathbb{R}$ is in this class, if there exists functions $g^{[i]} : S^{[i]} \subset \mathbb{R} \to \mathbb{R}$ and $h^{[i]} : S \subset \mathbb{R}^n \to \mathbb{R}$ with

$$\frac{\mathrm{d}f}{\mathrm{d}x_i}(x) = g^{[i]}(x_i) \cdot h^{[i]}(x). \tag{4}$$

Minimizer candidates can be obtained from $g^{[i]}(x_i) = 0$ which ensures first-order optimality condition

$$\frac{\mathrm{d}f}{\mathrm{d}x_i}(x) = 0,$$

is fulfilled. Minimizer that satisfy $h^{[i]}(x) = 0$ and minimizer at the boundary are not taken into consideration. Thus, there is no guarantee to obtain the global minimum by this approach.

In this paper, we aim to generalize the concept of separability in order to make previously non-decomposable functions also benefit from decomposition of the index set of the variables on subdomains that refer to the search space. Therefore, we introduce a special structure called structural separability and formulate an additional monotonicity condition on so-called structural separators in Sect. 2. Structural separability is less restrictive than (1), but has a similar property as (4). The monotonicity condition is necessary to guarantee that all possible optima are still considered after the decomposition, which is a crucial prerequisite for the integration into deterministic global optimization algorithms. Examples for functions that are

non-separable by (1) but fulfill the new definition such that their corresponding optimization problem can still be decomposed are given.

Section 3 explains how to implement the presented work and how to integrate it into a b&b algorithm for deterministic global optimization. Therefore, enclosures of the gradient (also: interval gradient) of the objective function as a combination of reliable and inclusion isotonic interval computations [12] and adjoint AD [13, 14] are used for the examination of the monotonicity condition and for automatic detection of separators. Interval gradients have already been used in deterministic global optimization for, among others, the verification of optimality conditions [4, 15–17]. But they can also be used for significance based approximate computing as presented in [18] and discussed in the context of neural networks in [19].

In Sect. 4, we show the results of a proof of concept implementation for the examples from Sect. 2 followed by conclusion and outlook in Sect. 5.

2 Subdomain separability

We introduce subdomain separability and we show how to exploit this property in deterministic global optimization.

Definition 1 (Structural separability) A function $f : S \subset \mathbb{R}^n \to \mathbb{R}$ is called structurally separable if there exists a decomposition of the index set of the variables into $p+1$ disjoint subsets $I^{[j]}$ such that

$$f(x) = h\left(f^{[1]}\left(x^{[1]}\right), \ldots, f^{[p]}\left(x^{[p]}\right), x^{[p+1]}\right), \; x^{[j]} \in S^{[j]}, \; \forall j, \tag{5}$$

with structural separators $f^{[j]} : S^{[j]} \subset \mathbb{R}^{n^{[j]}} \to \mathbb{R}$, $j \in \{1, \ldots, p\}$ and $h : \mathbb{R}^{p+n^{[p+1]}} \to \mathbb{R}$. Subsets $I^{[j]}$, $j \in \{1, \ldots, p\}$ are assumed to be non-empty, but $I^{[p+1]}$ can be empty if $p > 1$.

Remark 1 In the directed acyclic graph (DAG) representation of the optimization problem as described in [20], structural separators $f^{[j]}$ are cut-vertices [21], i.e., each path connecting vertices corresponding to variables $x^{[j]}$ with the vertex corresponding to the objective value $f(x)$ passes through the cut-vertex corresponding to $f^{[j]}$.

Remark 2 Conventionally separable functions as in (1) are covered by Definition 1 with

$$h\left(f^{[1]}\left(x^{[1]}\right), \ldots, f^{[p]}\left(x^{[p]}\right), \emptyset\right) = f^{[1]}\left(x^{[1]}\right) + \ldots + f^{[p]}\left(x^{[p]}\right).$$

Remark 3 Application of the chain rule of differentiation to differentiable structurally separable functions yields the gradient elements

$$\frac{\mathrm{d}f}{\mathrm{d}x^{[j]}}(x) = \frac{\mathrm{d}f}{\mathrm{d}f^{[j]}}(x) \cdot \frac{\mathrm{d}f^{[j]}}{\mathrm{d}x^{[j]}}\left(x^{[j]}\right), \; j \in \{1, \ldots, p\}, \tag{6}$$

with

$$\frac{\mathrm{d}f}{\mathrm{d}f^{[j]}}(x) = \frac{\partial h}{\partial f^{[j]}}\left(f^{[1]}(x^{[1]}), \ldots, f^{[p]}(x^{[p]}), x^{[p+1]}\right).$$

Equation (6) is the same as (4) with $g^{[j]}(x^{[j]}) = \frac{\mathrm{d}f^{[j]}}{\mathrm{d}x^{[j]}}(x^{[j]})$, $h^{[j]}(x) = \frac{\mathrm{d}f}{\mathrm{d}f^{[j]}}(x)$ and $n^{[j]} = 1, \forall j$.

Theorem 1 *(Subdomain separability) Consider the box constrained global optimization problem as in (2) with structurally separable and differentiable objective function f as in (5)*

and structural separators $f^{[j]}$, $j \in \{1, \ldots, p\}$. *If the objective function is monotonic w.r.t. a structural separator* $f^{[k]}$ *on the search space* S, *that is,*

$$\frac{\mathrm{d}f}{\mathrm{d}f^{[k]}}(x) \geq 0 \quad \forall x \in S \ \vee \ \frac{\mathrm{d}f}{\mathrm{d}f^{[k]}}(x) \leq 0 \quad \forall x \in S, \tag{7}$$

then the optimization problem in (2) can be decomposed into

$$\min_{x^{[j]} \in S^{[j]} \forall j \in \{1,\ldots,p+1\} \setminus k} h\left(f^{[1]}(x^{[1]}), \ldots, s^{[k],*}, \ldots, f^{[p]}(x^{[p]}), x^{[p+1]}\right), \tag{8}$$

$$\text{s.t. } s^{[k],*} = \begin{cases} \min\limits_{x^{[k]} \in S^{[k]}} f^{[k]}\left(x^{[k]}\right) \text{ if } \dfrac{\mathrm{d}f}{\mathrm{d}f^{[k]}}(x) \geq 0 \ \forall x \in S, \\ \max\limits_{x^{[k]} \in S^{[k]}} f^{[k]}\left(x^{[k]}\right) \text{ if } \dfrac{\mathrm{d}f}{\mathrm{d}f^{[k]}}(x) \leq 0 \ \forall x \in S. \end{cases} \tag{9}$$

Proof If the objective function f is monotonically increasing w.r.t. the structural separator $f^{[k]}$, i.e., $\frac{\mathrm{d}f}{\mathrm{d}f^{[k]}}(x) \geq 0 \ \forall x \in S$, we have

$$\begin{aligned} &h\left(f^{[1]}(x^{[1]}), \ldots, s^{[k],-}, \ldots, f^{[p]}(x^{[p]}), x^{[p+1]}\right) \\ &\quad \leq h\left(f^{[1]}(x^{[1]}), \ldots, s^{[k],+}, \ldots, f^{[p]}(x^{[p]}), x^{[p+1]}\right), \ \forall x^{[j]} \in S^{[j]}, \ \forall j, \end{aligned} \tag{10}$$

for $s^{[k],-} \leq s^{[k],+} \in \mathbb{R}$. As to $\frac{\partial h}{\partial x^{[k]}}(f^{[1]}(x^{[1]}), \ldots, f^{[p]}(x^{[p]}), x^{[p+1]}) = 0$, and due to (10) the global minimum of f requires the separator $f^{[k]}$ to be a minimum on the search space. The monotonic decrease scenario is handled analogously. □

Remark 4 The dimension of the inner optimization problem in (9) is $n^{[k]}$ while the dimension of the outer optimization problem in (8) is $n - n^{[k]} + 1$.

Remark 5 A degenerate solution is implied if $\frac{\mathrm{d}f}{\mathrm{d}f^{[k]}}(x) = 0, \ \forall x \in S$ and S contains more than one element.

Remark 6 If there are multiple structural separators, e.g., $f^{[j]}$ and $f^{[k]}$, $j \neq k$, fulfilling (7), then the inner optimization problems in (9) can be solved in parallel.

Remark 7 If structural separator $f^{[j]}$ is also structurally separable, then the decomposition approach can be applied recursively and the original optimization problem decomposes into even smaller disjoint optimization problems. In that case, we will introduce an additional superscript to distinguish structural separators belonging to different recursion depths. The j-th structural separator that belongs to recursion depth ℓ is denoted by $f^{[j],\ell}$.

If there are structural separators $f^{[j],\ell+1}$ and $f^{[k],\ell}$ with $I^{[j],\ell+1} \subset I^{[k],\ell}$, then the optimization problems for $f^{[j],\ell+1}$ should be solved before solving the optimization problems for $f^{[k],\ell}$.

Remark 8 If the monotonicity condition is violated, then the structural separability can still be exploited similar to [11] by solving $\frac{\mathrm{d}f^{[k]}}{\mathrm{d}x^{[k]}}\left(x^{[k]}\right) = 0$ for finding stationary points. As already stated in Sect. 1, this approach does not necessarily compute all stationary points.

For the remainder of this paper, we will consider that the evaluation formula of the objective function $y = f(x)$ is implemented as a differentiable program with *independent* variables

x and *dependent* variable y. Following [13], we assume that at a particular argument x the differentiable program of f can be expressed by a finite sequence of function evaluations as

$$\begin{aligned} v_i &= x_i, && i = 1, \ldots, n, \\ v_j &= \varphi_j(v_i)_{i \prec j}, && j = n+1, \ldots, n+p+1, \\ y &= v_{n+p+1}, \end{aligned} \tag{11}$$

where v_j for $j = n+1, \ldots, n+p$ are referred to as *intermediate* variables. The precedence relation $i \prec j$ indicates a direct dependency of v_j on v_i. Furthermore, the transitive closure $\prec^*$ of $\prec$ induces a partial ordering of all indices $j = 1, \ldots, n+p+1$. Equation (11) is also referred to as the single assignment code (SAC) of f. The SAC may not be unique due to commutativity, associativity and distributivity. We assume a SAC to be given.

2.1 Examples

Five test problems are investigated in the light of subdomain separability. They illustrate different aspects of the general approach. Besides the partially separable function in Example 1, there is the exponential function which is solvable in parallel and globally monotonic in Example 2, a recursive exponential function which is still globally monotonic but cannot be solved in parallel in Example 3 and the Shubert function in Example 4 that is not globally monotonic, but solvable in parallel. Example 5 can neither be solved in parallel nor it is globally monotonic but it could still benefit from subdomain separability. The SACs of the functions and their DAG representations are visualized to highlight that the vertices corresponding to the structural separators are cut-vertices.

Example 1 (Styblinski-Tang function [22]) Partially separable functions as in (1) are structurally separable and always fulfill the monotonicity condition in (7) with

$$\frac{\mathrm{d}f}{\mathrm{d}f^{[j]}}(x) = 1, \ \forall j \in \{1, \ldots, p\},$$

on any search space S which yields the well-known fact that the corresponding optimization problem can be decomposed and solved in parallel.

For example, the Styblinski-Tang function

$$f(x) = \frac{1}{2} \sum_{i=1}^{n} \left(x_i^4 - 16x_i^2 + 5x_i \right),$$

has the form of (1) except for the factor in front of the sum. Due to the factor, this function is marked as non-separable in [5]. Still, the problem can be decomposed into $f(x) = \frac{1}{2} \sum_{i=1}^{n} f^{[i]}(x^{[i]})$ with structural separators $f^{[i]} : \mathbb{R} \to \mathbb{R}$, $f^{[i]}(x^{[i]}) = x_i^4 - 16x_i^2 + 5x_i$ due to $\frac{\mathrm{d}f}{\mathrm{d}f^{[i]}}(x) = \frac{1}{2}, \ \forall i \in \{1, \ldots, n\}, \ \forall x \in \mathbb{R}^n$. The SAC of this function is given by

$$\begin{aligned} v_i &= x_i^4 - 16x_i^2 + 5x_i, \ i = 1, \ldots, n, \\ \Sigma &= \sum_{i=1}^{n} v_i, \\ y &= \frac{1}{2}\Sigma, \end{aligned} \tag{12}$$

and the DAG representing (12) is shown in Fig. 1.

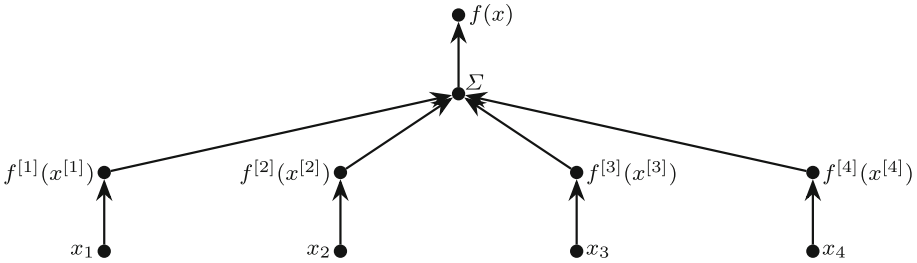

Fig. 1 DAG representing the program structure from Example 1 and Example 2 for $n = 4$

Example 2 (Exponential function [23]) For the exponential function

$$f(x) = -\exp\left(-\frac{1}{2}\sum_{i=1}^{n} x_i^2\right),$$

with SAC

$$\begin{aligned} v_i &= x_i^2, \ i = 1, \ldots, n, \\ \Sigma &= \sum_{i=1}^{n} v_i, \\ y &= -\exp\left(-\frac{1}{2}\Sigma\right), \end{aligned} \tag{13}$$

we have structural separators $f^{[i]} : \mathbb{R} \to \mathbb{R}$ with $f^{[i]}(x^{[i]}) = x_i^2$ and derivative of the objective w.r.t. these separators equal to

$$\frac{\mathrm{d}f}{\mathrm{d}f^{[i]}}(x) = \frac{1}{2}\exp\left(-\frac{1}{2}\sum_{i=1}^{n} x_i^2\right).$$

The exponential function is globally monotonically increasing such that Theorem 1 becomes applicable to all separators and the resulting subproblems can be solved in parallel.

The DAG of this function representing (13) is also given in Fig. 1 since it has the same structure as the DAG of the function from Example 1.

Example 3 (Recursive exponential function) For a demonstration of the usefulness of structural separability we consider the optimization problem in (2) with search space $S = [-2, 3]^n$, objective function $f : \mathbb{R}^n \to \mathbb{R}$,

$$\begin{aligned} f(x) &= h\left(f^{[1],1}(x^{[1],1}), x^{[2],1}\right), \\ f^{[1],\ell}\left(x^{[1],\ell}\right) &= h\left(f^{[1],\ell+1}(x^{[1],\ell+1}), x^{[2],\ell+1}\right), \ \forall \ell \in \{1, \ldots, n-2\}, \\ f^{[1],n-1}(v) &= \exp\left(v^2\right), \\ h(z, v) &= \exp\left(v^2 + z - 1\right), \end{aligned}$$

with structural separable function $h : \mathbb{R}^2 \to \mathbb{R}$, structural separators $f^{[1],\ell}$ and corresponding subsets of the index set $I^{[1],\ell} = \{1, \ldots, n-\ell\}$, $I^{[2],\ell} = \{n-\ell+1\}$, $\ell \in \{1, \ldots, n-1\}$.

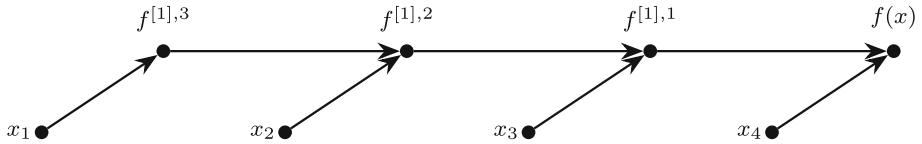

Fig. 2 DAG representing the program structure from Example 3 for $n = 4$

The differentiable program is given by the SAC

$$\begin{aligned} v_1 &= \exp(x_1), \\ v_i &= \exp(x_i^2 + v_{i-1} - 1), \ i = 2, \ldots, n, \\ y &= v_n. \end{aligned} \tag{14}$$

The program is non-decomposable in a conventional manner, but it fulfills Definition 1 for all structural separators. From the DAG representation of this function given in Fig. 2 it becomes clear that the separators are arranged in a row which requires a recursive decomposition of the optimization problem.

To decompose the optimization problem it remains to be shown that the derivatives of the objective w.r.t. the separators $\frac{\mathrm{d}f}{\mathrm{d}f^{[1],\ell}}(x), \ell \in \{1, \ldots, n-1\}$, are non-negative (or non-positive) on any subdomain. From

$$\frac{\mathrm{d}f}{\mathrm{d}f^{[1],1}}(x) = f(x),$$

and

$$\frac{\mathrm{d}f^{[1],\ell}}{\mathrm{d}f^{[1],\ell+1}}\left(x^{[1],\ell}\right) = f^{[1],\ell}\left(x^{[1],\ell}\right),$$

it follows that

$$\frac{\mathrm{d}f}{\mathrm{d}f^{[1],\ell}}(x) = \frac{\mathrm{d}f}{\mathrm{d}f^{[1],1}}(x) \cdot \prod_{j=1}^{\ell} \frac{\mathrm{d}f^{[1],j}}{\mathrm{d}f^{[1],j+1}}\left(x^{[1],j}\right) = f(x) \cdot \prod_{j=1}^{\ell} f^{[1],j}\left(x^{[1],j}\right).$$

By mathematical induction we show that $f^{[1],\ell}(x^{[1],\ell}) \geq 1$ for $\ell \in \{1, \ldots, n-1\}$. The basis $f^{[1],n-1}(x^{[1],n-1}) = \exp(x_1^2) \geq \exp(0) = 1$ obviously fulfills the statement. The assumption $f^{[1],\ell+1}(x^{[1],\ell+1}) \geq 1$ yields

$$\begin{aligned} f^{[1],\ell}\left(x^{[1],\ell}\right) &= \exp\left(x_{n-\ell+1}^2 + f^{[1],\ell+1}\left(x^{[1],\ell+1}\right) - 1\right) \\ &\geq \exp\left(x_{n-\ell+1}^2 + 1 - 1\right) \geq \exp(0) = 1, \end{aligned}$$

due to monotonicity of the exponential function. The same is true for $f(x)$. Thus, $f^{[1],\ell}(x^{[1],\ell}) \geq 1$ and $\frac{\mathrm{d}f}{\mathrm{d}f^{[1],\ell}}(x) \geq 1$ for $\ell \in \{1, \ldots, n-1\}$. Furthermore, we know that the global minimizer x^* is located at $x_i^* = 0, i \in \{1, \ldots, n\}$, with a minimum of $f(x^*) = 1$.

As a consequence of Theorem 1 the optimization problem can be reformulated as

$$\begin{aligned} &\min_{x_n \in [-2,3]} h\left(s^{[1],1,*}, x_n\right), \\ &\text{s.t. } s^{[1],\ell,*} = \min_{x_{n-\ell+1} \in [-2,3]} h\left(s^{[1],\ell+1,*}, x_{n-\ell+1}\right), \ \forall \ell \in \{1, \ldots, n-2\} \\ &\qquad s^{[1],n-1,*} = \min_{x_1 \in [-2,3]} \exp\left(x_1^2\right), \end{aligned}$$

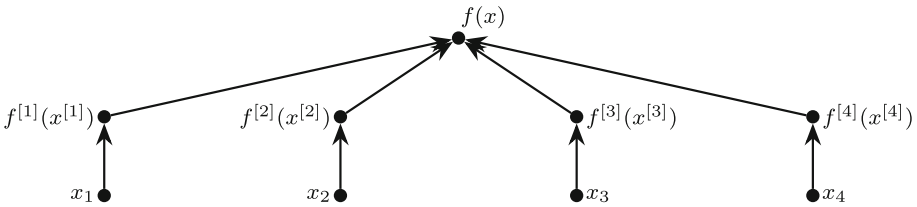

Fig. 3 DAG representing the program structure from Example 4 for $n = 4$

$$h(z, v) = \exp\left(v^2 + z - 1\right).$$

Note, that this function is globally monotonic w.r.t. the separator which does not necessarily hold in general. Since the separators are recursively dependent on each other, the corresponding optimization problems need to be solved sequentially beginning with $s^{[1],n-1,*}$.

Example 4 (Shubert function [24]) The Shubert function is given by $f : \mathbb{R}^n \to \mathbb{R}$ with

$$f(x) = \prod_{i=1}^{n} \sum_{j=1}^{5} \cos((j+1)x_i + j),$$

and SAC

$$\begin{aligned} v_i &= \sum_{j=1}^{5} \cos((j+1)x_i + j), \ i = 1, \ldots, n, \\ y &= \prod_{i=1}^{n} v_i, \end{aligned} \tag{15}$$

Its DAG representation is shown in Fig. 3. Each factor of the multiplication can be considered as a structural separator with $f^{[i]}(x^{[i]}) = \sum_{j=1}^{5} \cos((j+1)x_i + j)$ and $I^{[i]} = \{i\}$ for $i \in \{1, \ldots, n\}$. Derivatives of the function value w.r.t. the separators are derived as

$$\frac{\mathrm{d}f}{\mathrm{d}f^{[i]}}(x) = \prod_{\substack{k=1 \\ k \neq i}}^{n} f^{[k]}(x^{[k]}).$$

If the monotonicity condition holds for any structural separator, then the corresponding optimization problem can be decomposed by Theorem 1.

Example 5 (Salomon function [11]) We show that the Salomon function $f : \mathbb{R}^n \to \mathbb{R}$ is separable only on selected subdomains. The differentiable program is given by

$$f(x) = 1 - \cos\left(2\pi \sqrt{\sum_{i=1}^{n} x_i^2}\right) + 0.1 \sqrt{\sum_{i=1}^{n} x_i^2},$$

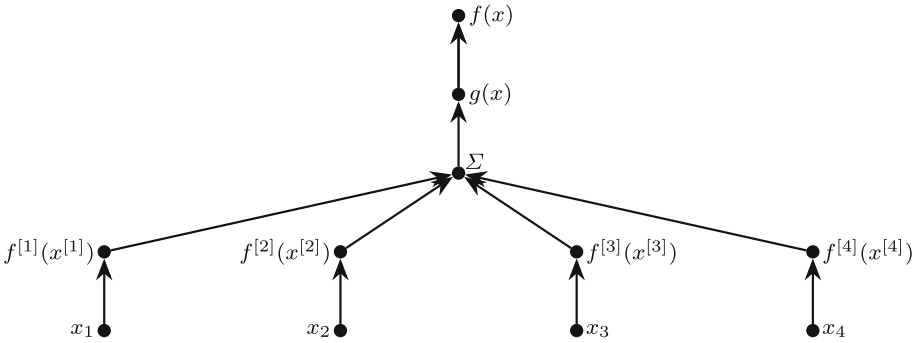

Fig. 4 DAG representing the structure of the evaluation formula from Example 5 for $n = 4$

with SAC

$$\begin{aligned} v_i &= x_i^2, \ i = 1, \ldots, n, \\ \Sigma &= \sum_{i=1}^{n} v_i, \\ g &= \sqrt{\Sigma}, \\ y &= 1 - \cos(2\pi g) + 0.1g. \end{aligned} \tag{16}$$

The DAG of (16) is visualized in Fig. 4. Introduction of an intermediate result $g : \mathbb{R}^n \to \mathbb{R}$ with $g(x) = \sqrt{\sum_{i=1}^{n} x_i^2}$ and structural separators $f^{[i]} : \mathbb{R} \to \mathbb{R}$ with $f^{[i]}(x^{[i]}) = x_i^2$, $i \in \{1, \ldots, n\}$, yields the derivatives

$$\begin{aligned} \frac{\mathrm{d}f}{\mathrm{d}f^{[i]}}(x) &= \frac{\mathrm{d}f}{\mathrm{d}g}(x) \cdot \frac{\mathrm{d}g}{\mathrm{d}f^{[i]}}(x), \\ \frac{\mathrm{d}f}{\mathrm{d}g}(x) &= 2\pi \sin(2\pi g(x)) + 0.1, \\ \frac{\mathrm{d}g}{\mathrm{d}f^{[i]}}(x) &= \frac{1}{2g(x)}. \end{aligned}$$

As $\frac{\mathrm{d}g}{\mathrm{d}f^{[i]}}(x)$ is always positive it remains to show that $\frac{\mathrm{d}f}{\mathrm{d}g}(x)$ is either non-positive (or non-negative) on any subdomain. The roots of $\frac{\mathrm{d}f}{\mathrm{d}g}(x)$ are

$$g_{2z-1} = z + \arcsin\left(-\frac{0.1}{2\pi}\right)\frac{1}{2\pi} \ \wedge \ g_{2z} = z - \frac{1}{2} - \arcsin\left(-\frac{0.1}{2\pi}\right)\frac{1}{2\pi}, \ z \in \mathbb{N}^+.$$

The function is monotonic between these roots. Thus, Theorem 1 can be applied to the Salomon function on the search domain $S = \left[\frac{g_z}{\sqrt{n}}, \frac{g_{z+1}}{\sqrt{n}}\right]^n$ for all $z \in \mathbb{N}^+$. If z is even, then the minimum of the separator is required for a minimum of the objective function. Otherwise, if z is odd, then the separator needs to be maximized to obtain a minimum of the objective function.

3 Implementation

In this section, we present how to compute interval gradients by adjoint AD, how they can be used to apply Theorem 1 to a differentiable program implementing a function f and how to verify structural separators.

3.1 Interval arithmetic

Interval arithmetic (IA) is a concept that enables the computation of bounds of a function evaluation on a given interval. A closed interval $[x] \in \mathbb{I}$ of a variable $x \in \mathbb{R}$ with lower bound $\underline{x} \in \mathbb{R}$ and upper bound $\overline{x} \in \mathbb{R}$ is denoted as

$$[x] = \left[\underline{x}, \overline{x}\right] = \left\{x \in \mathbb{R} \mid \underline{x} \leq x \leq \overline{x}\right\}.$$

If there is only a single element in $[x]$, i.e, the endpoints are equal $\underline{x} = \overline{x}$, then the square brackets $[\cdot]$ are dropped and x is called a degenerate interval. In that sense IA represents an extension of the real/floating-point number system.

Interval vectors $[x] \in \mathbb{I}^n$ have endpoints for each component

$$[x] = \left[\underline{x}, \overline{x}\right] = \left\{x \in \mathbb{R}^n \mid \underline{x_i} \leq x_i \leq \overline{x_i}\right\}.$$

When evaluating a function $y = f(x)$ in IA on $[x]$ we are interested in the information

$$[y] = f^*([x]) = \{f(x) \mid x \in [x]\}.$$

The asterisk denotes the united extension which computes the true range of values on the given domain. United extensions for all unary and binary elementary functions and arithmetic operations are known and endpoint formulas can be looked up, e.g., in [12]. Unfortunately, the derivation of endpoint formulas for the united extensions of composed functions might be expensive or even impossible. Hence, we will compute corresponding estimates by natural interval extensions. A natural interval extension can be obtained by replacing all elemental functions φ_j in (11) with their corresponding united extensions as

$$\begin{aligned} [v_i] &= [x_i], & i &= 1, \ldots, n, \\ \left[v_j\right] &= \varphi_j^*([v_i])_{i \prec j}, & j &= n+1, \ldots, n+p+1, \\ [y] &= \left[v_{n+p+1}\right]. \end{aligned} \tag{17}$$

The computation of the interval function value by the natural interval extension from (17) results in

$$[y] = f([x]) \supseteq f^*([x]).$$

The superset relation states that the interval $[y]$ can be an overestimation of all possible values over the given domain, but it guarantees enclosure. Furthermore, the natural interval extension of Lipschitz continuous functions converges linearly to the united extension with decreasing domain size.

The reader is referred to [12, 25] for more information on the topic.

3.2 Adjoint algorithmic differentiation

AD techniques [13, 14] use the chain rule to compute in addition to the function value of a computer program implementing the evaluation formula $f(x)$ of a function $f : \mathbb{R}^n \to \mathbb{R}$, its derivatives w.r.t. the independent variables x at a particular point x.

The adjoint or backward mode of AD propagates derivatives of the function w.r.t. independent and intermediate variables in reverse relative to the order of their computation in the SAC (11). The computationally intractable combinatorial optimization problem known as DAG REVERSAL [26] is implied.

Following [14], first-order adjoints are marked with a subscript $_{(1)}$. They are defined as

$$x_{(1)} = y_{(1)} \cdot \frac{\mathrm{d}f}{\mathrm{d}x}(x),$$

with $x_{(1)} \in \mathbb{R}^n$ and $y_{(1)} \in \mathbb{R}$. A single adjoint computation with seed $y_{(1)} = 1$ results in the gradient $\frac{\mathrm{d}f}{\mathrm{d}x}(x)$ stored in $x_{(1)}$. The adjoint of (11) can be implemented by (11) itself followed by

$$\begin{aligned} v_{(1),n+p+1} &= y_{(1)}, \\ v_{(1),k} &= \sum_{j:k\prec j} v_{(1),j} \cdot \frac{\partial \varphi_j}{\partial v_k}(v_i)_{i\prec j}, \quad k = n+p+1, \ldots, n+1, \\ x_{(1),i} &= v_{(1),i}, \qquad\qquad i = n, \ldots, 1. \end{aligned} \tag{18}$$

The evaluation of (18) yields the adjoints of all intermediate variables v_j in addition to those of the independent variables

$$v_{(1),j} = y_{(1)} \cdot \frac{\mathrm{d}f}{dv_j}(x), \quad j = n+p+1, \ldots, n+1.$$

3.3 Interval adjoints

The natural interval extension of (11) and (18) computes the interval function value and its interval derivatives w.r.t. all independent and intermediate variables as the result of a single evaluation. It can be implemented as (17) followed by

$$\begin{aligned} \left[v_{(1),n+p+1}\right] &= \left[y_{(1)}\right], \\ \left[v_{(1),k}\right] &= \sum_{j:k\prec j} \left[v_{(1),j}\right] \cdot \frac{\partial \varphi_j^*}{\partial v_k}([v_i])_{i\prec j}, \quad k = n+p+1, \ldots, n+1, \\ \left[x_{(1),i}\right] &= \left[v_{(1),i}\right], \qquad\qquad i = n, \ldots, 1. \end{aligned} \tag{19}$$

Compared to the traditional approach of AD in which the derivatives are only computed at specified points, we now get enclosures of the derivatives that contain all possible values of the derivative over the specified domain. The interval adjoints in (19) might be overestimated compared to the united extension as it is already stated for the interval values in Sect. 3.1. The natural interval extension of AD methods converges linearly for continuously differentiable functions [27]. Higher-order converging enclosures, e.g., centered forms [12], slopes or McCormick relaxations [28, 29] of AD methods can be derived [15, 27].

3.3.1 Monotonicity test

A single evaluation of the interval adjoint for $\left[y_{(1)}\right] = 1$ suffices to verify monotonicity as in (7) for all independent and intermediate variables. If the separation approach is embedded into a b&b solver that involves verification of the first-order optimality condition by interval adjoints, then the monotonicity test is for free, assuming that the separators are known a priori.

3.3.2 Verification of separators

Interval adjoints can be used to verify if an intermediate variable v is a structural separator of its program representing function f. W.l.o.g. we assume that the program is structurally separable with decomposition of the index set into two subsets $I^{[1]}$ and $I^{[2]}$, structural separator v dependent on $x^{[1]}$ and that its computing formula $v(x^{[1]})$ is implemented in the program. Furthermore, we assume that the interval adjoints $\frac{\mathrm{d}f}{\mathrm{d}v}([x]) \in \mathbb{I}$ as well as $\frac{\mathrm{d}f}{\mathrm{d}x^{[1]}}([x]) \in \mathbb{I}^{n^{[1]}}$ are already computed for the monotonicity test on $[x]$ as described in Sect. 3.3.1. Evaluation of (19) with the adjoint of the intermediate variable set to $\left[v_{(1)}\right] = \frac{\mathrm{d}f}{\mathrm{d}v}([x])$ yields

$$\left[x_{(1)}\right] = \frac{\mathrm{d}v}{\mathrm{d}x}\left(\left[x^{[1]}\right]\right) \cdot \frac{\mathrm{d}f}{\mathrm{d}v}([x]). \tag{20}$$

The assumption that f is structurally separable and satisfies Definition 1 with separator v is correct if

$$\begin{aligned} \frac{\mathrm{d}f}{\mathrm{d}x^{[1]}}(x) &= \frac{\mathrm{d}v}{\mathrm{d}x^{[1]}}\left(x^{[1]}\right) \cdot \frac{\mathrm{d}f}{\mathrm{d}v}(x), && \forall x \in [x], \\ \frac{\mathrm{d}v}{\mathrm{d}x^{[2]}}\left(x^{[1]}\right) &= 0, && \forall x \in [x], \end{aligned}$$

holds, which can be verified by checking equality

$$\left[x_{(1)}^{[1]}\right] = \frac{\mathrm{d}f}{\mathrm{d}x^{[1]}}([x]), \tag{21}$$

$$\left[x_{(1)}^{[2]}\right] = 0, \tag{22}$$

obtained from (20). If either (21) or (22) is violated, then v is not a structural separator. Thus, for the verification of each separator candidate one adjoint evaluation of (19) is required in addition to the one for the monotonicity test.

An exhaustive search for separators should be avoided, due to the potentially high number of intermediate variables and the associated number of separator candidates. Separators given by expert users can be verified efficiently. Since structural separability as given in Definition 1 is domain-independent and thus is a global property, it is sufficient to identify the separators once before performing the global search.

4 Case study

The general idea of b&b algorithms [25] used for global optimization problems as given in (2) is to remove all parts of the search space that cannot contain a global minimizer. The implementation used for this case study is a variation of the one presented in [17] implementing Theorem 1. The user needs to specify at least one separator. The algorithm performs the following steps:

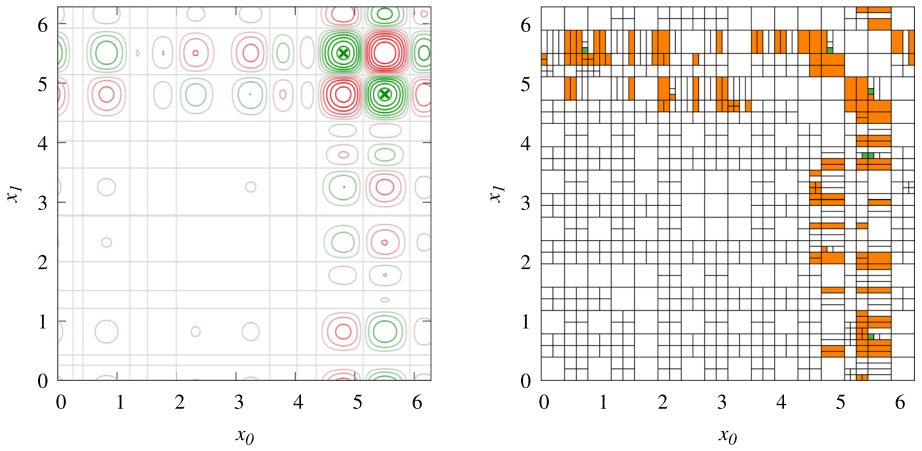

Fig. 5 Isolines of the Shubert function for $n = 2$ (left) with green lines around local minima and red lines around local maxima. B&b nodes considered by the algorithm (right) with active nodes marked in green, white boxes are discarded by the cut-off test and orange boxes if the first-order optimality condition is violated. Non-square boxes result from the index set decomposition approach

- *multi-section*: half-splitting in every dimension resulting in 2^n new b&b nodes;
- *cut-off test*: elimination of b&b node with search space $[x]$ if $f([x]) > \overline{y}^*$ with upper bound $\overline{y}^*$ for the global minimum;
- *first-order optimality test*: If $\frac{\mathrm{d}f}{\mathrm{d}x_i}([x]) \geq 0$ and $\underline{x_i}$ is a bound of original search space S, then recompute with $x_i = \underline{x_i}$, else if $\frac{\mathrm{d}f}{\mathrm{d}x_i}([x]) \leq 0$ and $\overline{x_i}$ is a bound of original search space S, then recompute with $x_i = \overline{x_i}$, otherwise eliminate b&b node with search space $[x]$;
- *improvement of bound* $\overline{y}^*$: Evaluate the function at any point (e.g., midpoint) of the search space to find a better bound $\overline{y}^*$;
- *separator test*: Check monotonicity condition for a priori known separators and generate a subproblem if Theorem 1 is applicable.

Obviously, the improvement of the upper bound of the global minimum can be enhanced by local searches instead of evaluation of the objective function at the midpoint of the current search space. Recursive separation is not supported by the current version of the solver. It is the subject of ongoing development efforts.

The software implements the required interval adjoints by using the interval type from the `Boost` library [30] as a base type of the first-order adjoint type provided by dco/c++[1] [31]. Both template libraries make use of the concept of operator overloading as supported, e.g., by C++.

On the left side of Fig. 5, isolines of the two-dimensional Shubert function over the domain $[0, 2\pi]$ are shown with green lines around (local) minima and red lines around local maxima. The two global minima are marked by green crosses. The right side of Fig. 5 shows the subdomains that are considered by the b&b algorithm. For visualization the branching is set up to stop at an accuracy of 0.1. Non-square search spaces result from the separation approach and only occur in regions that are proven to be monotonic by the interval adjoints. Green boxes are *active* search spaces that could contain a global minimizer. White boxes are discarded by the cut-off test. Orange boxes violate the first-order optimality condition.

[1] https://www.nag.co.uk/content/adjoint-algorithmic-differentiation

Table 1 Number of generated b&b nodes by the algorithm without and with separation

	n	Search space	w/o sep.	w/ sep.
Styblinski-Tang	4	$[-5, 5]^4$	4609	285
	8	$[-5, 5]^8$	5018817	569
Exponential	4	$[-1, 1]^4$	18	17
	8	$[-1, 1]^8$	258	33
Recursive exponential	4	$[-2.1, 2.0]^4$	273	252
	8	$[-2.1, 2.0]^8$	4609	549
Shubert	4	$[-10, 10]^4$	248618257	5272861
Salomon	4	$[-100, 100]^4$	2322	2322
	8	$[-100, 100]^8$	655618	655618

Our solver is used to find the global minima of the examples from Sect. 2. The algorithm is performed with and without separation. Structural separators are marked manually. The results are summarized in Table 1. Most of the presented examples benefit from the domain-dependent index set decomposition approach and have less b&b nodes generated by the algorithm if separation is enabled. The benefit increases with growing dimensionality due to the exponential complexity of the multi-section. The Salomon function does not benefit from the domain-dependent index set decomposition since the relevant domains are already discarded by the cut-off or first-order optimality tests.

We only measure runtimes for the Styblinski-Tang example with $n = 8$ with and without exploiting subdomain separability. Since the derivative information is already available for all separators after the first-order optimality test, the monotonicity test only iterates over the separators defined by the user. The number of b&b nodes considered by the algorithm without separation is 8820 times higher than with separation. The corresponding runtime without separation is only 7673 times higher than with separation. This observation correlates with the fact that the computations of b&b nodes that do not pass the cut-off test are terminated immediately. The percentage of b&b nodes that are eliminated due to the cut-off test is 30.2% for the case without separation and 2.8% with separation approach. The runtime estimates are averaged over 100 calls of the solver for both cases.

Our in-house solver has been designed as a playground for novel algorithms. Neither is it optimized for speed, nor does it feature state-of-the-art non-convex optimization methodology beyond the previously described b&b algorithm. Ultimately, we aim for integration of our ideas into modern software solutions for deterministic global optimization, e.g., [32, 33].

5 Conclusion and outlook

Our notion of separability combined with tests for monotonicity allows us to decompose an optimization problem into smaller optimization problems. It extends the verification of the first-order optimality condition as presented in the context of incomplete global search in [11] so that it can be used for deterministic global optimization, i.e., instead of just finding a few candidates fulfilling first-order optimality our approach guarantees to consider all possible optima.

We explained how to utilize interval adjoints to verify monotonicity of the objective function w.r.t. all structural separators at the cost of a single adjoint evaluation. As a first result, we revisited examples from the literature that benefit from the domain-dependent separability approach. Furthermore, we showed how to verify the separation property of a variable in a given computer program at the cost of only two adjoint evaluations.

The verification of separators can be used as a starting point for research into heuristics for automatically detecting separators in a computer program. Further work in progress includes enabling recursive separation. Moreover, interval arithmetic can result in a significant overestimation of the true value range, e.g., due to the wrapping effect or the dependency problem. Adjoint versions of interval centered forms, slopes and the McCormick relaxations should achieve better convergence.

Funding Open Access funding enabled and organized by Projekt DEAL.

References

1. Falk, J., Soland, R.: An algorithm for separable nonconvex programming problems. Manag. Sci. **15**, 550–569 (1969)
2. Horst, R., Tuy, H.: Global Optimization: Deterministic Approaches. Springer, Heidelberg (1990)
3. Floudas, C.A.: Deterministic Global Optimization: Theory. Methods and Applications. Springer, Boston, MA (2000)
4. Berenguel, J.L., Casado, L.G., García, I., Hendrix, E.M.T., Messine, F.: On interval branch-and-bound for additively separable functions with common variables. J. Glob. Optim. **56**, 1101–1121 (2013)
5. Jamil, M., Yang, X.: A literature survey of benchmark functions for global optimization problems. Int. J. Math. Model. Numer. Optim. **4**, 150–194 (2013)
6. Li, X., Tang, K., Omidvar, M.N., Yang, Z., Qin, K.: Benchmark functions for the CEC'2013 special session and competition on large-scale global optimization. Technical Report, Evolutionary Computation and Machine Learning Group, RMIT University, Australia. https://titan.csit.rmit.edu.au/~e46507/cec13-lsgo/competition/cec2013-lsgo-benchmark-tech-report.pdf (2013). Accessed 14 October 2020
7. Hadley, G.: Nonlinear and Dynamic Programming. Addison-Wesley, Boston, MA (1964)
8. Griewank, A., Toint, P.: On the unconstrained optimization of partially separable functions. In: Powell, M.J.D. (ed.) Nonlinear Optimization 1981, pp. 301–312. Academic Press, London (1982)
9. Phillips, A.T., Rosen, J.B.: A parallel algorithm for partially separable non-convex global minimization. Ann. Oper. Res. **25**, 101–118 (1990)
10. Gay, D.M.: Automatically finding and exploiting partially separable structure in nonlinear programming problems. Bell Laboratories, Murray Hill, NJ (1996)
11. Salomon, R.: Re-evaluating genetic algorithm performance under coordinate rotation of benchmark functions. A survey of some theoretical and practical aspects of genetic algorithms. BioSystems **39**, 263–278 (1996)
12. Moore, R.E., Kearfott, R.B., Cloud, M.J.: Introduction to Interval Analysis. SIAM, Philadelphia, PA (2009)
13. Griewank, A., Walther, A.: Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation. SIAM, Philadelphia, PA (2008)
14. Naumann, U.: The Art of Differentiating Computer Programs: An Introduction to Algorithmic Differentiation. SIAM, Philadelphia, PA (2012)
15. Schichl, H., Markót, M.C.: Algorithmic differentiation techniques for global optimization in the COCONUT environment. Optim. Methods Softw. **27**, 359–372 (2012)

16. Locatelli, M., Schoen, F.: Global Optimization: Theory, Algorithms, and Applications. SIAM, Philadelphia, PA (2013)
17. Deussen, J., Naumann, U.: Discrete interval adjoints in unconstrained global optimization. In: Le Thi, H., Le, H., Pham Dinh, T. (eds.) Optimization of Complex Systems: Theory, Models, Algorithms and Applications, pp. 78–88. Springer, Cham (2019)
18. Vassiliadis, V., Riehme, J., Deussen, J., Parasyris, K., Antonopoulos, C.D., Bellas, N., Lalis, S., Naumann, U.: Towards automatic significance analysis for approximate computing. In: Proceedings of CGO 2016 the 14th International Symposium on Code Generation and Optimization, pp. 182–193. ACM, New York, NY (2016)
19. Afghan, S., Naumann, U.: Interval adjoint significance analysis for neural networks. In: Krzhizhanovskaya, V.V., Závodszky, G., Lees, M.H., Dongarra, J.J., Sloot, P.M.A., Brissos, S., Teixeira, J. (eds.) Computational Science - ICCS 2020, pp. 365–378. Springer, Cham (2020)
20. Schichl, H., Neumaier, A.: Interval analysis on directed acyclic graphs for global optimization. J. Glob. Optim. **33**, 541–562 (2005)
21. Deo, N.: Graph Theory with Applications to Engineering and Computer Science. Prentice-Hall, Englewood Cliffs, NJ (1974)
22. Styblinski, M.A., Tang, T.S.: Experiments in nonconvex optimization: stochastic approximation with function smoothing and simulated annealing. Neural Netw. **3**, 467–483 (1990)
23. Rahnamyan, S., Tizhoosh, H., Salama, N.: Opposition-based differential evolution (ODE) with variable jumping rate. In: Proceedings of the 2007 IEEE Symposium on Foundations of Computational Intelligence, pp. 81–88. IEEE (2007)
24. Levy, A.V., Montalvo, A., Gomez, S., Calderon, A.: Topics in global optimization. In: Hennart, J.P. (ed.) Numerical Analysis. Proceedings of the Third IIMAS Workshop, Lect. Notes Math. 909, pp. 18–33. Springer, Berlin (1982)
25. Hansen, E., Walster, G.W.: Global Optimization Using Interval Analysis. Marcel Dekker, New York, NY (2004)
26. Naumann, U.: DAG reversal is NP-complete. J. Discrete Algorithms **7**, 402–410 (2009)
27. Deussen, J.: Global Derivatives. PhD thesis. RWTH Aachen University (2021)
28. McCormick, G.P.: Computability of global solutions to factorable nonconvex programs: part I—convex underestimating problems. Math. Program. **10**, 147–175 (1976)
29. Mitsos, A., Chachuat, B., Barton, P.: McCormick-based relaxation of algorithms. SIAM J. Optim. **20**, 573–601 (2009)
30. Brönnimann, H., Melquiond, G., Pion, S.: The design of the Boost interval arithmetic library. Theor. Comput. Sci. **351**, 111–118 (2006)
31. Naumann, U., Leppkes, K., Lotz, J.: Derivative code by overloading in C++ (dco/c++): Introduction and summary of features. Technical Report, Aachener Informatik Berichte (AIB-2016-08), RWTH Aachen University, Aachen. http://aib.informatik.rwth-aachen.de/2016/2016-08.pdf (2016). Accessed 14 October 2020
32. Sahinidis, N.V.: BARON: a general purpose global optimization software package. J. Glob. Optim. **8**, 201–205 (1996)
33. Bongartz, D., Najman, J., Sass, S., Mitsos, A.: MAiNGO–McCormick-based algorithm for mixed-integer nonlinear global optimization. Technical Report, Process Systems Engineering (AVT.SVT), RWTH Aachen University, Aachen. http://www.avt.rwth-aachen.de/global/show_document.asp?id=aaaaaaaaabclahw (2018). Accessed 14 October 2020