J Glob Optim (2015) 63:1–36
DOI 10.1007/s10898-015-0303-6

Reverse propagation of McCormick relaxations

Achim Wechsung[1] · **Joseph K. Scott**[2] · **Harry A. J. Watson**[1] · **Paul I. Barton**[1]

Received: 2 October 2013 / Accepted: 12 April 2015 / Published online: 23 April 2015

Abstract Constraint propagation techniques have heavily utilized interval arithmetic while the application of convex and concave relaxations has been mostly restricted to the domain of global optimization. Here, reverse McCormick propagation, a method to construct and improve McCormick relaxations using a directed acyclic graph representation of the constraints, is proposed. In particular, this allows the interpretation of constraints as implicitly defining set-valued mappings between variables, and allows the construction and improvement of relaxations of these mappings. Reverse McCormick propagation yields potentially tighter enclosures of the solutions of constraint satisfaction problems than reverse interval propagation. Ultimately, the relaxations of the objective of a non-convex program can be improved by incorporating information about the constraints.

Keywords Constraint propagation · Global optimization · Convex relaxations · McCormick relaxations

Mathematics Subject Classification 49M20 · 49M37 · 65K05 · 90C26

✉ Paul I. Barton
pib@mit.edu

Achim Wechsung
awechsun@mit.edu

Joseph K. Scott
jks9@clemson.edu

Harry A. J. Watson
hwatson@mit.edu

1 Process Systems Engineering Laboratory, Massachusetts Institute of Technology, 77 Massachusetts Ave, Rm 66-363, Cambridge, MA 02139, USA

2 Department of Chemical and Biomolecular Engineering, Clemson University, 207B Earle Hall, Clemson, SC 29634, USA

1 Introduction

A constraint satisfaction problem (CSP) consists of a finite set of variables, domains and constraints. A solution of a CSP is an assignment of elements of the domains to the variables so that all constraints are satisfied. In general, these problems are NP-hard and hence it is desirable to compute an enclosure of the solution set. Constraint propagation routines, or, more generally, contractors, are numerical methods that assist in this task. Using the information about the relationship between variables that is contained in a single constraint or in a set of constraints, they attempt to shrink an initial enclosure of the domains. Typically, intervals are used to enclose the solution sets whereas a constraint propagation technique for McCormick relaxations [39,52] is proposed in this contribution that is applicable to factorable functions.

1.1 Review of constraint propagation methods

Constraint propagation was first developed for logic constraints on discrete domains [35]. Different notions of consistency, which describes the degree to which the remaining elements of the domain satisfy the constraints, have been introduced for this case [2,9]. Constraint propagation has also been applied to connected sets that appear in so-called numerical CSPs [8,13] and a large number of techniques have been proposed in the literature.

Many constraint propagation methods use ideas from interval analysis: they consider interval domains and use interval arithmetic. Cleary [12] and Davis [13] presented the first algorithms for constraint propagation with interval domains. Hyvönen [26] considered cases where exact numbers are insufficient and studied how interval arithmetic can be utilized in CSPs. Lhomme [33] proposed an extension of arc-consistency to numeric CSPs. Benhamou et al. [6] introduced the notion of box-consistency. Sam-Haroud and Faltings [49] approximated feasible regions by 2^n-trees and presented algorithms to label leaves consistently. Benhamou and Older [5] proposed the notion of hull-consistency. Van Hentenryck et al. [59] showed how interval extensions can be used to calculate box-consistent labels, see also [60]. Benhamou et al. [7] proposed an algorithm for hull-consistency that does not require decomposing constraints into primitives. Vu et al. [61] proposed a method to construct inner and outer approximations of the feasible set using unions of intervals. Lebbah et al. [32] discussed how the reformulation-linearization technique can be used to relax nonlinear constraints and to aid in pruning the search space. Granvilliers and Benhamou [19] proposed an algorithm that prunes boxes using both constraint propagation techniques and the interval Newton method. Recently, Domes and Neumaier [15] proposed a constraint propagation method for linear and quadratic constraints and constraints and Jaulin [27] studied set-valued CSPs.

Jaulin et al. [28] discussed contractors based on interval analysis, many of which were also the subject of Neumaier's book, though it focused on solving systems of equations in the presence of data uncertainty [43]. Recently, Schichl and Neumaier [50] studied directed acyclic graphs (DAG) to represent functions for interval evaluation. Vu et al. [62] used this representation and extended the contractor proposed in [7], which propagates interval information forward and backward along the DAG. Recently, Stuber et al. [55] extended contractors based on interval analysis to compute convex and concave relaxations of implicit functions. However, their methods require existence and uniqueness of the implicit function on the full domain.

1.2 Connection to global optimization

Continuous optimization problems are often solved to guaranteed global optimality using continuous branch-and-bound algorithms [17,25]. It is well-known that the efficiency of these algorithms can be improved by discarding parts of the search space that are infeasible or that are known not to contain optimal solutions [56]. These tasks are often referred to as *domain reduction*. Obviously, global optimization is an important application of CSPs [44] and ideas originally developed for CSPs are routinely utilized in global optimization: logic-based methods can enhance and expedite optimization routines [24], constraint propagation is often used to discard parts of the domain where the solution is known not to exist (e.g., [21, 22,47]). For example, constraint propagation routines are part of BARON's pre-processing step [48]. It is also not coincidental that many constraint satisfaction tools use branch-and-prune frameworks inspired by global optimization algorithms to identify a set of boxes that contains all solutions (e.g., [19,32,59]). Also, see the recent discussion of feasibility-based bounds-tightening procedures in [3,4]. Thus, borrowing and embracing ideas from the other field has been very beneficial for both fields.

As indicated by the "bound" keyword, branch-and-bound algorithms also require computable bounds on the range of the objective function and non-convex constraints. Interval methods have been used to provide such bounds [42,43,45,56]. However, their slow convergence results in long computational times when the number of variables is large, which led to the development of several nonlinear convex relaxation techniques [1,39,41,52,56,57] that are more accurate and have higher convergence order [10]. Methods for domain reduction, and CSPs in general, however, still rely almost exclusively on interval arithmetic.

1.3 Replacing intervals with relaxations in constraint propagation

Schichl and Neumaier [50] demonstrated that factorable functions can be represented alternatively as a DAG[1] and discussed how this representation can be used for calculations in interval analysis. Vu et al. [62] detailed how to propagate interval information on DAGs to improve interval bounds. Their method can utilize the information from equality and inequality constraints. We will refer to this idea as *reverse interval propagation*. In this paper, the idea is extended to convex and concave relaxations.

The remainder of the paper is organized as follows. In Sect. 2, the proposed method will be described briefly. In Sect. 3, the notion of a factorable function is defined and concepts from interval and McCormick analysis are reviewed. Section 4 recapitulates the important results for reverse interval propagation from [62], which are extended to McCormick objects in Sect. 5. Section 6 discusses how the theoretical results from the previous section can be applied to construct, improve and utilize relaxations in the context of CSPs and global optimization. Section 7 describes how the method can be implemented and some small illustrative examples are given in Sect. 8. The results of applying the method to a set of more complicated global optimization test problems are shown in Sect. 9. Section 10 summarizes the contributions and concludes the paper.

2 Method description

In this section, we summarize the proposed method and describe how it can be applied in the context of solving constraint satisfaction problems and optimization problems globally.

[1] This representation of a factorable function is also used in the reverse mode of automatic differentiation [20].

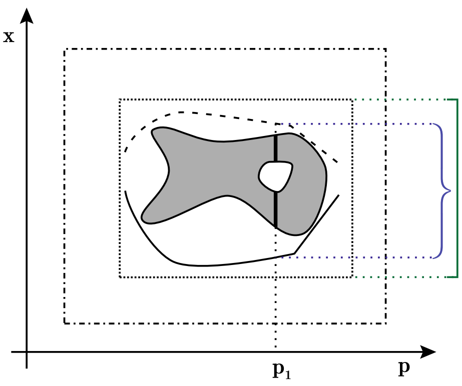

Fig. 1 Illustration of domain reduction by reverse interval and McCormick propagation. The *gray area* is the set of all feasible points, the *dash-dotted line* is the original domain, the *dotted line* is the reduced domain using reverse interval propagation. The *solid* and *dashed lines* are relaxations of the feasible region that are functions of p

The class of factorable functions encompasses most functions that can be implemented as computer programs without conditional statements. It is well-known that relaxations of factorable functions can be computed using McCormick's composition rule [39,52]; the obtained relaxations are often referred to as *McCormick relaxations*. Here, it is proposed to use the DAG representation of the constraints to also propagate McCormick relaxations backward. For the benefit of the reader we provide an interpretation of relaxations in the context of constraint propagation. Suppose we partition the variables into p and x. Whereas reverse interval propagation yields a constant interval bound that all feasible (x, p) must satisfy, reverse McCormick propagation yields bounds that are (convex and concave) functions of p. For a given p in the domain, all x so that (p, x) is feasible are bounded. Figure 1 illustrates this interpretation. It shows that a domain (dash-dotted box) can be shrunk by interval constraint propagation to find an outer approximation of the feasible region (dotted box). However, the relaxations (solid and dashed lines) can provide a tighter approximation that is a function of p. For example, consider p_1, for which a thick solid line shows all feasible x. Given p_1, the relaxations restrict x to the interval (curly brace in Fig. 1) whereas the interval bounds only constrain them to the larger interval (square bracket). Furthermore, since the bounds are convex and concave functions of p, it is tractable, for example, to calculate a reduced interval domain using affine relaxations based on the subgradients of the relaxations [41] or by minimizing and maximizing the relaxations of each x_i on the p domain.

When solving global optimization problems, the improved domains for x and p can be used as input to the calculation of the relaxations of the objective function. By taking advantage of the information about the feasible region, it is possible to improve the tightness of the objective function relaxations, a very desirable feature in global optimization.

The method can be described as follows: First, a particular partitioning of the variables is selected and initial interval bounds $(\boldsymbol{p}, \boldsymbol{x})$ on the variables are specified. For each variable suitable initial values are then derived from these bounds. Next, for each factor of the function F bounds and McCormick relaxations are computed. After this forward pass, bounds as well as convex and concave relaxations of the reachable set $\{F(x, p) : (x, p) \in \boldsymbol{x} \times \boldsymbol{p}\}$ have been constructed. Now, known restrictions of this reachable set, i.e., equality or inequality constraints are used to tighten these bounds and relaxations. Lastly, this information is propagated back to the variables x and p by "inverting", in some sense, the operation related to each factor of the function. This yields the relaxations of the feasible space described above.

3 Preliminary definitions and results

In this section, factorable functions will be formally defined with the following development in mind. The notation follows [51, Chapter 2] closely. Also, some concepts from interval and McCormick analysis are reviewed. In particular, Sect. 3.3 utilizes many definitions introduced in [51, Chapter 2].

3.1 Factorable functions

Loosely speaking, a function is factorable if it can be represented as a finite sequence of simple binary operations and univariate functions.

Herein, a function will be denoted as a triple (o, B, R) where B is the domain, R is the range, and o is a mapping from B into R, $o : B \to R$. Permissible functions shall include binary addition $(+, \mathbb{R}^2, \mathbb{R})$ and multiplication $(\times, \mathbb{R}^2, \mathbb{R})$ as well as a collection of univariate functions, cf. Definition 1.

Definition 1 Let $\mathscr{L}$ denote a set of functions $(u, B, \mathbb{R})$ where $B \subset \mathbb{R}$. $\mathscr{L}$ will be referred to as a *library of univariate functions*.

It will be required that, for each $(u, B, \mathbb{R}) \in \mathscr{L}$, $u(x)$ can be evaluated on a computer for any $x \in B$. Additional assumptions will be introduced when necessary.

Factorable functions will be defined in terms of *computational sequences*, which are ordered sequences of the permissible basic operations defined above. Every such sequence of computations defines a sequence of intermediate quantities called *factors*. In the following definition, the factors are denoted by v_k, and the functions π_k are used to select one or two previous factors to be the operand(s) of the next operation. Note that a computational sequence is a specialization of a DAG because it allows binary and unary operations only.

Definition 2 Let $n_i, n_o \geq 1$. A $\mathscr{L}$*-computational sequence* with n_i inputs and n_o outputs is a pair $(\mathscr{S}, \pi_o)$, where:

1. $\mathscr{S}$ is a finite sequence of pairs $\{((o_k, B_k, \mathbb{R}), (\pi_k, \mathbb{R}^{k-1}, \mathbb{R}^{d_k}))\}_{k=n_i+1}^{n_f}$ with every element defined by one of the following options:
   (a) $(o_k, B_k, \mathbb{R})$ is either $(+, \mathbb{R}^2, \mathbb{R})$ or $(\times, \mathbb{R}^2, \mathbb{R})$ and $\pi_k : \mathbb{R}^{k-1} \to \mathbb{R}^2$ is defined by $\pi_k(v) = (v_i, v_j)$ for some integers $i, j \in \{1, \ldots, k-1\}$.
   (b) $(o_k, B_k, \mathbb{R}) \in \mathscr{L}$ and $\pi_k : \mathbb{R}^{k-1} \to \mathbb{R}$ is defined by $\pi_k(v) = v_i$ for some integer $i \in \{1, \ldots, k-1\}$.
2. $\pi_o : \mathbb{R}^{n_f} \to \mathbb{R}^{n_o}$ is defined by $\pi_o(v) = (v_{i(1)}, \ldots, v_{i(n_o)})$ for some integers $i(1), \ldots, i(n_o) \in \{1, \ldots, n_f\}$.

A computational sequence defines a function $F_{\mathscr{S}} : D_{\mathscr{S}} \subset \mathbb{R}^{n_i} \to \mathbb{R}^{n_o}$ by the following construction.

Definition 3 Let $(\mathscr{S}, \pi_o)$ be a $\mathscr{L}$-computational sequence with n_i inputs and n_o outputs. Define the *sequence of factors* $\{(v_k, D_k, \mathbb{R})\}_{k=1}^{n_f}$ with $D_k \subset \mathbb{R}^{n_i}$, where

1. for $k = 1, \ldots, n_i$, $D_k = \mathbb{R}^{n_i}$ and $v_k(x) = x_k$, $\forall x \in D_k$,
2. for $k = n_i + 1, \ldots, n_f$, $D_k = \{x \in D_{k-1} : \pi_k(v_1(x), \ldots, v_{k-1}(x)) \in B_k\}$ and $v_k(x) = o_k(\pi_k(v_1(x), \ldots, v_{k-1}(x)))$, $\forall x \in D_k$.

The set $D_{\mathscr{S}} \equiv D_{n_f}$ is the *natural domain* of $(\mathscr{S}, \pi_o)$, and the *natural function* $(F_{\mathscr{S}}, D_{\mathscr{S}}, \mathbb{R}^{n_o})$ is defined by $F_{\mathscr{S}}(x) = \pi_o(v_1(x), \ldots, v_{n_f}(x))$, $\forall x \in D_{\mathscr{S}}$.

Definition 4 A function $F : D \subset \mathbb{R}^n \to \mathbb{R}^m$ is *$\mathscr{L}$-factorable* if there exists a $\mathscr{L}$-computational sequence $(\mathscr{S}, \pi_o)$ with n inputs and m outputs such that the natural function $(F_{\mathscr{S}}, D_{\mathscr{S}}, \mathbb{R}^m)$ satisfies $D \subset D_{\mathscr{S}}$ and $F = F_{\mathscr{S}}|_D$.

3.2 Interval analysis

Definition 5 We conform to the standardized interval notation outlined in [30]. For $a, b \in \mathbb{R}$, $a \leq b$ define the *interval* $[a, b]$ as the compact, connected set $\{x \in \mathbb{R} : a \leq x \leq b\}$. The set of all nonempty intervals is denoted as $\mathbb{IR}$, and intervals are denoted by bold face letters, $\boldsymbol{x} \in \mathbb{IR}$. The set of n-dimensional boxes (Cartesian products of n intervals) is denoted by $\mathbb{IR}^n$. The "interval vector" notation $(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n)$ will often be used for $\boldsymbol{x}_1 \times \boldsymbol{x}_2 \times \cdots \times \boldsymbol{x}_n$. Suppose $\boldsymbol{x} \in \mathbb{IR}^n$. Then, the *lower* and *upper bounds* of $\boldsymbol{x}$ are denoted as $\underline{x}$ and $\overline{x}$, respectively. Suppose $Z \subset \mathbb{R}^n$. The set of all interval subsets of Z is denoted by $\mathbb{I}Z \subset \mathbb{IR}^n$. Lastly, if Z is nonempty and bounded, then $\square Z$ with $(\square Z)_i = [\inf_{z \in Z} z_i, \sup_{z \in Z} z_i]$, $i = 1, \ldots, n$ denotes the *interval hull* of Z, the tightest box enclosing Z. Note that $(\cdot)^L$ and $(\cdot)^U$ will be used for more complex expressions to denote the respective lower and upper bound vectors of a box.

We will encounter functions that either return a vector of reals or the symbol NaN, or "not a number", which can be thought of as undefined or unspecified. It is convenient to define $\mathbb{R}_{\emptyset} = \mathbb{R} \cup \{\text{NaN}\}$. We also define ${}^*\mathbb{R} = \mathbb{R} \cup \{-\infty, +\infty\}$ to denote the extended reals. For the purposes of this paper it is also necessary to extend the definition of an interval to include unbounded intervals and empty intervals, which are commonly excluded in the definition of $\mathbb{IR}$ (e.g, [30]). Here, $\emptyset$ is used to denote the empty interval.

Definition 6 Let $\mathbb{I}_{\emptyset}\mathbb{R} \equiv \mathbb{IR} \cup \{\emptyset\}$, and let the set of all interval subsets of $Z \subset \mathbb{R}^n$ including the empty interval be denoted by $\mathbb{I}_{\emptyset}Z \subset \mathbb{I}_{\emptyset}\mathbb{R}^n$. Similarly to Definition 5, define the *set of all extended intervals* as ${}^*\mathbb{IR} = \{[a, b] : a \in \mathbb{R} \cup \{-\infty\}, b \in \mathbb{R} \cup \{+\infty\}, a \leq b\} \cup \{\emptyset\}$, which includes all unbounded intervals and also the empty interval. Lastly, the set of all extended interval subsets of $Z \subset \mathbb{R}^n$ is denoted by ${}^*\mathbb{I}Z \subset {}^*\mathbb{IR}^n$.

We will follow the conventions that real-valued operations involving NaN result in NaN, that $[\text{NaN}, \text{NaN}] = \emptyset$, that NaN is an element of any interval, that every interval $\boldsymbol{x} \in \mathbb{I}_{\emptyset}\mathbb{R}$ or $\boldsymbol{x} \in {}^*\mathbb{IR}$ contains the empty interval and that any interval operation involving the empty interval will again result in the empty interval with the exception of the construction of the interval hull where $\square\{\boldsymbol{x}, \emptyset\} = \boldsymbol{x}$ for any $\boldsymbol{x} \in {}^*\mathbb{IR}^n$. Note that $\boldsymbol{x} = \emptyset$ for $\boldsymbol{x} \in {}^*\mathbb{IR}^n$ if $\boldsymbol{x}_i = \emptyset$ for some $i = 1, \ldots, n$. Otherwise, the operations of interval arithmetic extend naturally. For any $x \in \mathbb{R}$ and $\circ \in \{+, -, \cdot, /\}$, define $x \circ \pm\infty = \lim_{a \to \pm\infty} x \circ a$.

As described in detail in Sect. 6.1 one benefit of this definition is the ease with which potential domain violations occurring during an evaluation of the natural function can be handled. If we let points outside the natural domain evaluate to NaN which, by our convention, is an element of any interval then the all-important inclusion property of interval functions, which we will define below, can be maintained.

Definition 7 Let $F : D \subset \mathbb{R}^n \to \mathbb{R}^m_{\emptyset}$, and for any $\boldsymbol{x} \subset D$, let range$(F, \boldsymbol{x})$ denote the image of the box $\boldsymbol{x}$ under F. A mapping $\boldsymbol{F} : \mathfrak{D} \subset {}^*\mathbb{I}D \to {}^*\mathbb{IR}^m$ is an *inclusion function* for F on $\mathfrak{D}$ if range$(F, \boldsymbol{x}) \subset \boldsymbol{F}(\boldsymbol{x})$, $\forall \boldsymbol{x} \in \mathfrak{D}$.

Definition 8 Let $D \subset \mathbb{R}^n$. A set $\mathfrak{D} \subset {}^*\mathbb{IR}^n$ is an *interval extension* of D if $\mathfrak{D} \subset {}^*\mathbb{I}D$ and every $x \in D$ satisfies $[x, x] \in \mathfrak{D}$. Let $F : D \to \mathbb{R}^m_{\emptyset}$. A function $\boldsymbol{F} : \mathfrak{D} \subset {}^*\mathbb{I}D \to {}^*\mathbb{IR}^m$ is an *interval extension* of F on D if $\mathfrak{D}$ is an interval extension of D and $\boldsymbol{F}([x, x]) = [F(x), F(x)]$ for every $x \in D$.

Definition 9 Let $\boldsymbol{F} : \mathfrak{D} \subset {}^*\mathbb{IR}^n \to {}^*\mathbb{IR}^m$. $\boldsymbol{F}$ is *inclusion monotonic* on $\mathfrak{D}$ if $\boldsymbol{x}_1 \subset \boldsymbol{x}_2$ implies that $\boldsymbol{F}(\boldsymbol{x}_1) \subset \boldsymbol{F}(\boldsymbol{x}_2), \forall \boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathfrak{D}$.

Theorem 1 *Let $F : D \subset \mathbb{R}^n \to \mathbb{R}^m_\emptyset$ and let $\boldsymbol{F} : \mathfrak{D} \to {}^*\mathbb{IR}^m$ be an interval extension of F. If $\boldsymbol{F}$ is inclusion monotonic on $\mathfrak{D}$, then $\boldsymbol{F}$ is an inclusion function for F on $\mathfrak{D}$.*

Proof Choose any $\boldsymbol{x} \in \mathfrak{D}$ and any $x \in \boldsymbol{x}$. Since $x \in D$, it follows that $[x, x] \in \mathfrak{D}$. Since $\emptyset \in \boldsymbol{F}(\boldsymbol{x})$ is always true, if $f_i(x) = \text{NaN}$ for some $i \in \{1, \ldots, m\}$ then $F(x) \in \boldsymbol{F}(\boldsymbol{x})$. Otherwise, the result follows from [51, Theorem 2.3.4]. □

Define the typical inclusion functions for addition and multiplication: let the functions $(+, \mathbb{I}_\emptyset\mathbb{R}^2, \mathbb{I}_\emptyset\mathbb{R})$ and $(\times, \mathbb{I}_\emptyset\mathbb{R}^2, \mathbb{I}_\emptyset\mathbb{R})$ be defined by $+(\boldsymbol{x}, \boldsymbol{y}) \equiv [\underline{x} + \underline{y}, \overline{x} + \overline{y}]$ and $\times(\boldsymbol{x}, \boldsymbol{y}) \equiv [\min(\underline{x}\,\underline{y}, \underline{x}\,\overline{y}, \overline{x}\,\underline{y}, \overline{x}\,\overline{y}), \max(\underline{x}\,\underline{y}, \underline{x}\,\overline{y}, \overline{x}\,\underline{y}, \overline{x}\,\overline{y})]$ and recall our convention[2] that any operation involving the empty interval results in an empty interval, i.e., $+(\boldsymbol{x}, \emptyset) = +(\emptyset, \boldsymbol{x}) = \emptyset$ or $\times(\boldsymbol{x}, \emptyset) = \times(\emptyset, \boldsymbol{x}) = \emptyset$ for any $\boldsymbol{x} \in \mathbb{I}_\emptyset\mathbb{R}$.

Assumption 1 Assume that for every $(u, B, \mathbb{R}) \in \mathscr{L}$, an interval extension $(u, \mathbb{I}_\emptyset B, \mathbb{I}_\emptyset\mathbb{R})$ is known. Furthermore, this interval extension is inclusion monotonic on $\mathbb{I}_\emptyset B$.

Suppose that Assumption 1 holds and that $(\mathscr{S}, \pi_o)$ is a $\mathscr{L}$-computational sequence. Then, to any element (o_k, π_k) of $\mathscr{S}$ a corresponding $(o_k, \mathbb{I}_\emptyset B_k, \mathbb{I}_\emptyset\mathbb{R})$ exists. Also, the functions $(\pi_k, \mathbb{I}_\emptyset\mathbb{R}^{k-1}, \mathbb{I}_\emptyset\mathbb{R})$ or $(\pi_k, \mathbb{I}_\emptyset\mathbb{R}^{k-1}, \mathbb{I}_\emptyset\mathbb{R}^2)$ with $\pi_k(\boldsymbol{v}) = (\boldsymbol{v}_i)$ or $\pi_k(\boldsymbol{v}) = (\boldsymbol{v}_i, \boldsymbol{v}_j)$ extend $(\pi_k, \mathbb{R}^{k-1}, \mathbb{R})$ or $(\pi_k, \mathbb{R}^{k-1}, \mathbb{R}^2)$ naturally. Note that we do not distinguish notationally between the real and interval versions of the elementary operations and functions u, π_k, and o_k. Rather, they are assumed to always act on the class of the object in their argument(s).

Definition 10 For every $\mathscr{L}$-computational sequence $(\mathscr{S}, \pi_o)$ with n_i inputs and n_o outputs, define the *sequence of inclusion factors* $\{(\boldsymbol{v}_k, \mathfrak{D}_k, \mathbb{I}_\emptyset\mathbb{R})\}_{k=1}^{n_f}$ where

1. for all $k = 1, \ldots, n_i$, $\mathfrak{D}_k = \mathbb{I}_\emptyset\mathbb{R}^{n_i}$ and $\boldsymbol{v}_k(\boldsymbol{x}) = \boldsymbol{x}_k, \forall \boldsymbol{x} \in \mathfrak{D}_k$,
2. for all $k = n_i + 1, \ldots, n_f$, $\mathfrak{D}_k = \{\boldsymbol{x} \in \mathfrak{D}_{k-1} : \pi_k(\boldsymbol{v}_1(\boldsymbol{x}), \ldots, \boldsymbol{v}_{k-1}(\boldsymbol{x})) \in \mathbb{I}_\emptyset B_k\}$ and $\boldsymbol{v}_k(\boldsymbol{x}) = o_k(\pi_k(\boldsymbol{v}_1(\boldsymbol{x}), \ldots, \boldsymbol{v}_{k-1}(\boldsymbol{x}))), \forall \boldsymbol{x} \in \mathfrak{D}_k$.

The *natural interval extension* of $(\mathscr{S}, \pi_o)$ is the function $(\boldsymbol{F}_\mathscr{S}, \mathfrak{D}_\mathscr{S}, \mathbb{I}_\emptyset\mathbb{R}^{n_o})$ defined by $\mathfrak{D}_\mathscr{S} \equiv \mathfrak{D}_{n_f}$ and $\boldsymbol{F}_\mathscr{S}(\boldsymbol{x}) = \pi_o(\boldsymbol{v}_1(\boldsymbol{x}), \ldots, \boldsymbol{v}_{n_f}(\boldsymbol{x})), \forall \boldsymbol{x} \in \mathfrak{D}_\mathscr{S}$.

Theorem 2 *Let $(\mathscr{S}, \pi_o)$ be a $\mathscr{L}$-computational sequence with associated natural function $(F_\mathscr{S}, D_\mathscr{S}, \mathbb{R}^{n_o})$. The natural interval extension $(\boldsymbol{F}_\mathscr{S}, \mathfrak{D}_\mathscr{S}, \mathbb{I}_\emptyset\mathbb{R}^{n_o})$ is an inclusion monotonic interval extension of $(F_\mathscr{S}, D_\mathscr{S}, \mathbb{R}^{n_o})$ on $\mathfrak{D}_\mathscr{S}$ and an inclusion function for $F_\mathscr{S}$ on $\mathfrak{D}_\mathscr{S}$. In particular, each inclusion factor $\boldsymbol{v}_k$ of $(\mathscr{S}, \pi_o)$ is an inclusion monotonic interval extension of v_k on $\mathfrak{D}_\mathscr{S}$ for all $k = 1, \ldots, n_f$.*

Proof See [51, Theorem 2.3.11] together with Theorem 1. □

3.3 McCormick analysis

Let $D \subset \mathbb{R}^n$ be convex. A vector-valued function $G : D \to \mathbb{R}^m$ is *convex* on D if each component is convex on D. Similarly, it is *concave* on D if each component is concave on D. For any set A, let $\mathbb{P}(A)$ denote the *power set*, or set of all subsets, including the empty set, of A.

[2] Hereafter, we will not make this distinction explicitly in expressions. Rather it is always assumed tacitly.

Definition 11 Let $D \subset \mathbb{R}^n$ be a convex set and $F : D \to \mathbb{P}(\mathbb{R}^m)$. A function $\check{F} : D \to \mathbb{R}^m$ is a *convex relaxation*, or *convex underestimator*, of F on D if $\check{F}$ is convex on D and $\check{f}_i(x) \leq \inf\{f_i(x)\}, \forall x \in D$ and $i = 1, \ldots, m$. A convex relaxation $G : D \to \mathbb{R}^m$ is called the *convex envelope* of F on D if $g_i(x) \geq \check{f}_i(x)$ for all convex relaxations of F, $\forall x \in D$ and $i = 1, \ldots, m$. A function $\hat{F} : D \to \mathbb{R}^m$ is a *concave relaxation*, or *concave overestimator*, of F on D if $\hat{F}$ is concave on D and $\hat{f}_i(x) \geq \sup\{f_i(x)\}, \forall x \in D$ and $i = 1, \ldots, m$. A concave relaxation $G : D \to \mathbb{R}^m$ is called the *concave envelope* of F on D if $g_i(x) \leq \hat{f}_i(x)$ for all concave relaxations of F, $\forall x \in D$ and $i = 1, \ldots, m$.

Remark 1 Definition 11 allows that $f_i(x) = \text{NaN}$ for some $i \in \{1, \ldots, m\}$ and $x \in D$. In this case, the inequality defining a relaxation will hold for any function. However, the convexity and concavity requirement must still be met by $\check{F}$ and $\hat{F}$, respectively, and this requirement constrains the set of functions that satisfy the definition, as exemplified in Fig. 7.

The following notation was introduced in [51]. While it differs from the previously used notation for McCormick relaxations, it is more compact and very useful for the proposed operations on computational sequences, and it also makes the relationship with interval arithmetic more apparent. In the latter, information is passed from one operation in the sequence of factors to the next in the form of intervals. McCormick's procedure to construct relaxations [39], on the other hand, requires a box $\boldsymbol{x}$ and a point $x \in \boldsymbol{x}$ as input and returns three values: a box $\boldsymbol{v}_k(\boldsymbol{x})$, which encloses the image of $\boldsymbol{x}$ under v_k, and two additional values $\check{v}_k(\boldsymbol{x}, x)$ and $\hat{v}_k(\boldsymbol{x}, x)$, which represent the value of the convex and concave relaxation of v_k on $\boldsymbol{x}$ evaluated at x. After a recent generalization, one can also consider mappings that take a box and two relaxation values as input and return a box and two relaxation values; these are called generalized McCormick relaxations [52]. One advantage of this generalization is that it yields mappings with conformable inputs and outputs, which are hence composable.

Definition 12 Let $\mathbb{MR}^n \equiv \{(\boldsymbol{z}^B, \boldsymbol{z}^C) \in \mathbb{IR}^n \times \mathbb{IR}^n : \boldsymbol{z}^B \cap \boldsymbol{z}^C \neq \emptyset\}$. Elements of $\mathbb{MR}^n$ are denoted by script capitals, $\mathscr{Z} \in \mathbb{MR}^n$. For any $\mathscr{Z} \in \mathbb{MR}^n$, the notations $\boldsymbol{z}^B, \boldsymbol{z}^C \in \mathbb{IR}^n$ and $(\underline{z}, \overline{z}, \check{z}, \hat{z}) \in \mathbb{R}^n$ will be commonly used to denote the boxes and vectors satisfying $\mathscr{Z} = (\boldsymbol{z}^B, \boldsymbol{z}^C) = ([\underline{z}, \overline{z}], [\check{z}, \hat{z}])$. For any $D \subset \mathbb{R}^n$, let $\mathbb{M}D$ denote the set $\{\mathscr{Z} \in \mathbb{MR}^n : \boldsymbol{z}^B \subset D\}$. Note that for more complex expressions, $(\cdot)^C$ will be used to denote the relaxation box, and $(\cdot)^{cv}$ and $(\cdot)^{cc}$ will be used to denote the convex and concave relaxation vectors, respectively, of a McCormick object.

In this paper, it is also necessary to consider unbounded and empty McCormick objects. Analogous to Definition 6, define the sets $\mathbb{M}_\emptyset\mathbb{R}^n \equiv \{(\boldsymbol{z}^B, \boldsymbol{z}^C) \in \mathbb{I}_\emptyset\mathbb{R}^n \times \mathbb{I}_\emptyset\mathbb{R}^n : \boldsymbol{z}^B \cap \boldsymbol{z}^C \neq \emptyset \vee \boldsymbol{z}^C = \emptyset\}$ and ${}^*\mathbb{MR}^n \equiv \{(\boldsymbol{z}^B, \boldsymbol{z}^C) \in {}^*\mathbb{IR}^n \times {}^*\mathbb{IR}^n : \boldsymbol{z}^B \cap \boldsymbol{z}^C \neq \emptyset \vee \boldsymbol{z}^C = \emptyset\}$, which are extensions of $\mathbb{MR}^n$. Also, define $\mathbb{M}_\emptyset D$ and ${}^*\mathbb{M}D$ for any $D \in \mathbb{R}^n$ analogous to $\mathbb{I}_\emptyset D$ and ${}^*\mathbb{I}D$. Introduce $\text{Enc} : {}^*\mathbb{MR}^n \to {}^*\mathbb{IR}^n$ defined by $\text{Enc}(\mathscr{Z}) = \boldsymbol{z}^B \cap \boldsymbol{z}^C$ for all $\mathscr{Z} \in {}^*\mathbb{MR}^n$. This function is necessary since for $z \in \mathbb{R}^n_\emptyset$, $z \in \mathscr{Z}$ is not well-defined whereas $z \in \text{Enc}(\mathscr{Z})$ is.

Next, we formalize McCormick's technique by defining operations on $\mathbb{M}_\emptyset\mathbb{R}^n$. We introduce the concept of a *relaxation function*, which is analogous to the notion of an inclusion function in interval analysis, and is the fundamental object that we want to compute for a given real-valued function. Then, we show how relaxation functions can be obtained through a simpler construction, just as inclusion functions can be constructed from inclusion monotonic interval extensions. First, however, some preliminary concepts are necessary.

Definition 13 Let $\mathscr{X}, \mathscr{Y} \in {}^*\mathbb{MR}^n$. $\mathscr{X}$ and $\mathscr{Y}$ are *coherent* if $\boldsymbol{x}^B = \boldsymbol{y}^B$. A set $\mathscr{D} \subset {}^*\mathbb{MR}^n$ is coherent if every pair of elements is coherent. A set $\mathscr{D} \subset {}^*\mathbb{MR}^n$ is *closed under coherence* if, for any coherent $\mathscr{X}, \mathscr{Y} \in {}^*\mathbb{MR}^n$, $\mathscr{X} \in \mathscr{D}$ implies $\mathscr{Y} \in \mathscr{D}$.

For any coherent $\mathscr{X}_1, \mathscr{X}_2 \in {}^*\mathbb{MR}^n$ with a common box part $\boldsymbol{q}$ and for all $\lambda \in [0, 1]$, define

$$\mathrm{Conv}(\lambda, \mathscr{X}_1, \mathscr{X}_2) \equiv (\boldsymbol{q}, \lambda \boldsymbol{x}_1^C + (1-\lambda)\boldsymbol{x}_2^C)$$

where the rules of interval arithmetic are used to evaluate $\lambda \boldsymbol{x}_1^C + (1-\lambda)\boldsymbol{x}_2^C$. For any $\mathscr{X}_1, \mathscr{X}_2 \in {}^*\mathbb{MR}^n$, the inclusion $\mathscr{X}_1 \subset \mathscr{X}_2$ holds iff $\boldsymbol{x}_1^B \subset \boldsymbol{x}_2^B$ and $\boldsymbol{x}_1^C \subset \boldsymbol{x}_2^C$. Likewise, $\mathscr{X}_1 \supset \mathscr{X}_2$ iff $\mathscr{X}_2 \subset \mathscr{X}_1$. Also, define $\mathscr{X}_1 \cap \mathscr{X}_2 \equiv (\boldsymbol{x}_1^B \cap \boldsymbol{x}_2^B, \boldsymbol{x}_1^C \cap \boldsymbol{x}_2^C)$.

Definition 14 Suppose $\mathscr{D} \subset {}^*\mathbb{MR}^n$ is closed under coherence. A function $\mathscr{F} : \mathscr{D} \to {}^*\mathbb{MR}^m$ is *coherently concave* on $\mathscr{D}$ if for every coherent $\mathscr{X}_1, \mathscr{X}_2 \in \mathscr{D}$, i.e., $\boldsymbol{q} = \boldsymbol{x}_1^B = \boldsymbol{x}_2^B$, $\mathscr{F}(\mathscr{X}_1)$ and $\mathscr{F}(\mathscr{X}_2)$ are coherent, and $\mathscr{F}(\mathrm{Conv}(\lambda, \mathscr{X}_1, \mathscr{X}_2)) \supset \mathrm{Conv}(\lambda, \mathscr{F}(\mathscr{X}_1), \mathscr{F}(\mathscr{X}_2))$ for every $\lambda \in [0, 1]$.

Definition 15 Let $F : D \subset \mathbb{R}^n \to \mathbb{R}^m_\emptyset$. A mapping $\mathscr{F} : \mathscr{D} \subset {}^*\mathbb{M}D \to {}^*\mathbb{MR}^m$ is a *relaxation function* for F on $\mathscr{D}$ if $\mathscr{D}$ is closed under coherence, $\mathscr{F}$ is coherently concave on $\mathscr{D}$, and $F(x) \in \mathrm{Enc}(\mathscr{F}(\mathscr{X}))$ is satisfied for every $\mathscr{X} \in \mathscr{D}$ and $x \in \mathrm{Enc}(\mathscr{X})$.

Remark 2 Definition 14 is a generalization of convexity and concavity, and Definition 15 is a generalization of the notion of convex and concave relaxations. Convex and concave relaxations of F can be recovered from a relaxation function of F as follows. Let $\boldsymbol{x} \in \mathbb{I}D$ so that there exists $\mathscr{Y} \in \mathscr{D}$ with $\boldsymbol{x} = \boldsymbol{y}^B$. Define the functions $\mathscr{U}, \mathscr{O} : \boldsymbol{x} \to \mathbb{R}^m_\emptyset$ for all $x \in \boldsymbol{x}$ by $([\underline{F}, \overline{F}], [\mathscr{U}(x), \mathscr{O}(x)]) \equiv \mathscr{F}((\boldsymbol{x}, [x, x]))$. Then, $\mathscr{U}$ and $\mathscr{O}$ are convex and concave relaxations of F on $\boldsymbol{x}$, respectively, as shown in [51, Lemma 2.4.11].

Definition 16 Let $D \subset \mathbb{R}^n$. A set $\mathscr{D} \subset {}^*\mathbb{MR}^n$ is a *McCormick extension* of D if $\mathscr{D} \subset {}^*\mathbb{M}D$ and every $x \in D$ satisfies $([x, x], [x, x]) \in \mathscr{D}$. Let $F : D \to \mathbb{R}^m_\emptyset$. A function $\mathscr{F} : \mathscr{D} \to {}^*\mathbb{MR}^m$ is a *McCormick extension* of F if $\mathscr{D}$ is a McCormick extension of D and $\mathscr{F}([x, x], [x, x]) = ([F(x), F(x)], [F(x), F(x)]), \forall x \in D$.

Definition 17 Let $\mathscr{F} : \mathscr{D} \subset {}^*\mathbb{MR}^n \to {}^*\mathbb{MR}^m$. $\mathscr{F}$ is *inclusion monotonic* on $\mathscr{D}$ if $\mathscr{X}_1 \subset \mathscr{X}_2$ implies that $\mathscr{F}(\mathscr{X}_1) \subset \mathscr{F}(\mathscr{X}_2)$ for all $\mathscr{X}_1, \mathscr{X}_2 \in \mathscr{D}$.

Theorem 3 *Let $F : D \subset \mathbb{R}^n \to \mathbb{R}^m_\emptyset$ and let $\mathscr{F} : \mathscr{D} \to {}^*\mathbb{MR}^m$ be a McCormick extension of F. If $\mathscr{F}$ is inclusion monotonic on $\mathscr{D}$, then every $\mathscr{X} \in \mathscr{D}$ satisfies $F(x) \in \mathrm{Enc}(\mathscr{F}(\mathscr{X}))$ for all $x \in \mathrm{Enc}(\mathscr{X})$.*

Proof See [51, Theorem 2.4.14]. □

We conclude that an inclusion monotonic McCormick extension that is also coherently concave is a relaxation function. Hence, it suffices to derive an inclusion monotonic, coherently concave McCormick extension. As shown in [51, Lemmas 2.4.15,2.4.17] the composition of inclusion monotonic, coherently concave McCormick extensions yields an inclusion monotonic, coherently concave McCormick extension. This motivates the derivations of inclusion monotonic, coherently concave McCormick extensions of the basic operations below.

Define the following relaxation functions for addition and multiplication: let the functions $(+, \mathbb{M}_\emptyset\mathbb{R}^2, \mathbb{M}_\emptyset\mathbb{R})$ and $(\times, \mathbb{M}_\emptyset\mathbb{R}^2, \mathbb{M}_\emptyset\mathbb{R})$ be given by $+(\mathscr{X}, \mathscr{Y}) = (\boldsymbol{x}^B + \boldsymbol{x}^B, \boldsymbol{x}^C + \boldsymbol{x}^C)$ and $\times(\mathscr{X}, \mathscr{Y}) = (\boldsymbol{x}^B \boldsymbol{y}^B, [\check{z}, \hat{z}])$ where

$$\check{z} = \max\left(\left(\underline{y}\boldsymbol{x}^C + \underline{x}\boldsymbol{y}^C - \underline{x}\,\underline{y}\right)^L, \left(\overline{y}\boldsymbol{x}^C + \overline{x}\boldsymbol{y}^C - \overline{x}\,\overline{y}\right)^L, (\boldsymbol{x}^B\boldsymbol{y}^B)^L\right),$$

$$\hat{z} = \min\left(\left(\underline{y}\boldsymbol{x}^C + \overline{x}\boldsymbol{y}^C - \overline{x}\,\underline{y}\right)^U, \left(\overline{y}\boldsymbol{x}^C + \underline{x}\boldsymbol{y}^C - \underline{x}\,\overline{y}\right)^U, (\boldsymbol{x}^B\boldsymbol{y}^B)^U\right).$$

Note that this definition of multiplication ensures that $[\check{z}, \hat{z}] \subset \boldsymbol{x}^B \boldsymbol{y}^B$ [51, Theorems 2.4.22, 2.4.23]. Furthermore, the standard rules for addition and multiplication of McCormick relaxations are implied by these definitions, see [51, p. 69f], with the addition of the intersection with the bounds from interval arithmetic in the case of the multiplication rule. These functions are indeed relaxation functions and also inclusion monotonic as shown in [51, Section 2.4.2].

The following assumption is needed to construct relaxation functions for the elements of $\mathscr{L}$. For many univariate functions, objects satisfying these assumptions are known and readily available [51, Section 2.8].

Assumption 2 Assume that for every $(u, B, \mathbb{R}) \in \mathscr{L}$, functions $\check{u}, \hat{u} : \tilde{B} \to \mathbb{R}$ where $\tilde{B} \equiv \{(\boldsymbol{x}, x) \in \mathbb{I}B \times B : x \in \boldsymbol{x}\}$ and $x^{\min}, x^{\max} : \mathbb{I}B \to \mathbb{R}$ are known such that $\check{u}(\boldsymbol{x}, \cdot)$ and $\hat{u}(\boldsymbol{x}, \cdot)$ are convex and concave relaxations of u on $\boldsymbol{x} \in \mathbb{I}B$, respectively, and $x^{\min}(\boldsymbol{x})$ and $x^{\max}(\boldsymbol{x})$ are a minimum of $\check{u}(\boldsymbol{x}, \cdot)$ and a maximum of $\hat{u}(\boldsymbol{x}, \cdot)$ on $\boldsymbol{x}$, respectively. Furthermore, assume that for any $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{I}B$ with $\boldsymbol{x}_1 \subset \boldsymbol{x}_2$, $\check{u}(\boldsymbol{x}_1, x) \geq \check{u}(\boldsymbol{x}_2, x)$ and $\hat{u}(\boldsymbol{x}_1, x) \leq \hat{u}(\boldsymbol{x}_2, x)$ for all $x \in \boldsymbol{x}_1$ and that $\check{u}([x, x], x) = \hat{u}([x, x], x)$ for all $x \in B$.

Let mid : $\mathbb{R} \times \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ return the middle value of its arguments. It can be shown (cf. [51], p. 76f) that a relaxation function of $(u, B, R) \in \mathscr{L}$ is given by $(u, \mathbb{M}B, \mathbb{M}\mathbb{R})$ with

$$u(\mathscr{X}) = \Big(u(\boldsymbol{x}^B), \Big[\check{u}(\boldsymbol{x}^B, \operatorname{mid}(\check{x}, \hat{x}, x^{\min}(\boldsymbol{x}^B))), \hat{u}(\boldsymbol{x}^B, \operatorname{mid}(\check{x}, \hat{x}, x^{\max}(\boldsymbol{x}^B)))\Big] \cap u(\boldsymbol{x}^B)\Big). \tag{1}$$

Note that if the convex and concave envelopes of u are known and used, then the intersection with $u(\boldsymbol{x}^B)$ in (1) is redundant.

Suppose that Assumption 2 holds and that $(\mathscr{S}, \pi_o)$ is a $\mathscr{L}$-computational sequence. Then, for any element (o_k, π_k) of $\mathscr{S}$, the preceding developments provide an inclusion monotonic McCormick extension $(o_k, \mathbb{M}_\emptyset B_k, \mathbb{M}_\emptyset \mathbb{R})$. Also, the functions $(\pi_k, \mathbb{M}_\emptyset \mathbb{R}^{k-1}, \mathbb{M}_\emptyset \mathbb{R})$ or $(\pi_k, \mathbb{M}_\emptyset \mathbb{R}^{k-1}, \mathbb{M}_\emptyset \mathbb{R}^2)$ with $\pi_k(\mathscr{V}) = (\mathscr{V}_i)$ or $\pi_k(\mathscr{V}) = (\mathscr{V}_i, \mathscr{V}_j)$ extend $(\pi_k, \mathbb{R}^{k-1}, \mathbb{R})$ or $(\pi_k, \mathbb{R}^{k-1}, \mathbb{R}^2)$ naturally.

Definition 18 For every $\mathscr{L}$-computational sequence $(\mathscr{S}, \pi_o)$ with n_i inputs and n_o outputs, define the *sequence of relaxation factors* $\{(\mathscr{V}_k, \mathscr{D}_k, \mathbb{M}_\emptyset \mathbb{R})\}_{k=1}^{n_f}$ where

1. for all $k = 1, \ldots, n_i$, $\mathscr{D}_k = \mathbb{M}_\emptyset \mathbb{R}^{n_i}$ and $\mathscr{V}_k(\mathscr{X}) = \mathscr{X}_k, \forall \mathscr{X} \in \mathscr{D}_k$,
2. for all $k = n_i + 1, \ldots, n_f$, $\mathscr{D}_k = \{\mathscr{X} \in \mathscr{D}_{k-1} : \pi_k(\mathscr{V}_1(\mathscr{X}), \ldots, \mathscr{V}_{k-1}(\mathscr{X})) \in \mathbb{M}_\emptyset B_k\}$ and $\mathscr{V}_k(\mathscr{X}) = o_k(\pi_k(\mathscr{V}_1(\mathscr{X}), \ldots, \mathscr{V}_{k-1}(\mathscr{X}))), \forall \mathscr{X} \in \mathscr{D}_k$.

The *natural McCormick extension* of $(\mathscr{S}, \pi_o)$ is the function $(\mathscr{F}_\mathscr{S}, \mathscr{D}_\mathscr{S}, \mathbb{M}\mathbb{R}^{n_o})$ defined by $\mathscr{D}_\mathscr{S} \equiv \mathscr{D}_{n_f}$ and $\mathscr{F}_\mathscr{S}(\mathscr{X}) = \pi_o(\mathscr{V}_1(\mathscr{X}), \ldots, \mathscr{V}_{n_f}(\mathscr{X})), \forall \mathscr{X} \in \mathscr{D}_\mathscr{S}$.

Theorem 4 *Let $(\mathscr{S}, \pi_o)$ be a $\mathscr{L}$-computational sequence with associated natural function $(F_\mathscr{S}, D_\mathscr{S}, \mathbb{R}^{n_o})$. The natural McCormick extension $(\mathscr{F}_\mathscr{S}, \mathscr{D}_\mathscr{S}, \mathbb{M}_\emptyset \mathbb{R}^{n_o})$ is a McCormick extension of $(F_\mathscr{S}, D_\mathscr{S}, \mathbb{R}^{n_o})$ and coherently concave and inclusion monotonic on $\mathscr{D}_\mathscr{S}$. Thus, it is a relaxation function for $F_\mathscr{S}$ on $\mathscr{D}_\mathscr{S}$. In particular, each relaxation factor $\mathscr{V}_k$ of $(\mathscr{S}, \pi_o)$ is a inclusion monotonic, coherently concave McCormick extension of v_k on $\mathfrak{D}_\mathscr{S}$ for all $k = 1, \ldots, n_f$.*

Proof See [51, Theorem 2.4.32] together with Theorem 3. □

Thus, so far we have described forward propagation of intervals and McCormick objects, as is commonly done to compute natural interval extensions and standard McCormick relaxations. Next, we consider reverse propagation of intervals and describe its use in CSPs. The formal development of forward McCormick propagation in this section as an analogous process to forward interval propagation allows us to then extend the reverse interval propagation ideas to McCormick objects in Sect. 5.

4 Reverse interval propagation

In this section, we will focus on propagating interval bounds backwards through the computational sequence, which is a particular form of a DAG. Since the reverse McCormick propagation is similar in spirit, it is very instructive to first revisit the interval case. The results, which are stated below, have been adapted from [62], though the notation is introduced here.

Definition 19 Consider $F : D \subset \mathbb{R}^n \to \mathbb{R}^m$. Let $\boldsymbol{F}^{\text{rev}} : \mathbb{I}_{\emptyset} D \times {}^*\mathbb{IR}^m \to \mathbb{I}_{\emptyset}\mathbb{R}^n$. If for all $\boldsymbol{x} \in \mathbb{I}_{\emptyset} D$ and $\boldsymbol{r} \in {}^*\mathbb{IR}^m$ it holds that

$$\{x \in \boldsymbol{x} : F(x) \in \boldsymbol{r}\} \subset \{x \in \boldsymbol{F}^{\text{rev}}(\boldsymbol{x}, \boldsymbol{r})\} \subset \boldsymbol{x}, \tag{2}$$

then $\boldsymbol{F}^{\text{rev}}$ is called a *reverse interval update* of F.

Definition 20 Let $(\mathscr{S}, \pi_o)$ be a $\mathscr{L}$-computational sequence with n_i inputs and n_o outputs with natural interval extension $(\boldsymbol{F}_{\mathscr{S}}, \mathfrak{D}_{\mathscr{S}}, \mathbb{R}^{n_o})$. Let $\boldsymbol{x} \in \mathfrak{D}_{\mathscr{S}}$. Suppose that $\boldsymbol{v}_1(\boldsymbol{x}), \ldots, \boldsymbol{v}_{n_f}(\boldsymbol{x})$ have been calculated according to Definition 10. Let $o_k^{\text{rev}} : \mathbb{I}_{\emptyset} B_k \times {}^*\mathbb{IR} \to \mathbb{I}_{\emptyset} B_k$ be a reverse interval update of o_k for each $k = n_i + 1, \ldots, n_f$. Suppose $\tilde{\boldsymbol{v}}_1, \ldots, \tilde{\boldsymbol{v}}_{n_f} \in \mathbb{I}_{\emptyset}\mathbb{R}$ are calculated for any $\boldsymbol{x} \in \mathfrak{D}_{\mathscr{S}}$ and $\boldsymbol{r} \in {}^*\mathbb{IR}^{n_o}$ by the following procedure:

```
$(\tilde{\boldsymbol{v}}_1, \ldots, \tilde{\boldsymbol{v}}_{n_f}) := (\boldsymbol{v}_1(\boldsymbol{x}), \ldots, \boldsymbol{v}_{n_f}(\boldsymbol{x}))$
$\pi_o(\tilde{\boldsymbol{v}}_1, \ldots, \tilde{\boldsymbol{v}}_{n_f}) := \pi_o(\tilde{\boldsymbol{v}}_1, \ldots, \tilde{\boldsymbol{v}}_{n_f}) \cap \boldsymbol{r}$
for $l := 1, \ldots, n_f - n_i$ do
    $\pi_{n_f-l+1}(\tilde{\boldsymbol{v}}_1, \ldots, \tilde{\boldsymbol{v}}_{n_f-l}) := o_{n_f-l+1}^{\text{rev}}(\pi_{n_f-l+1}(\tilde{\boldsymbol{v}}_1, \ldots, \tilde{\boldsymbol{v}}_{n_f-l}), \tilde{\boldsymbol{v}}_{n_f-l+1})$
end
```

The *reverse interval propagation* of $(\mathscr{S}, \pi_o)$ is the function $(\boldsymbol{F}_{\mathscr{S}}^{\text{rev}}, \mathfrak{D}_{\mathscr{S}} \times {}^*\mathbb{IR}^{n_o}, \mathbb{I}D_{\mathscr{S}})$ defined by $\boldsymbol{F}_{\mathscr{S}}^{\text{rev}}(\boldsymbol{x}, \boldsymbol{r}) \equiv \tilde{\boldsymbol{v}}_1 \times \cdots \times \tilde{\boldsymbol{v}}_{n_i}$.

Theorem 5 *The reverse interval propagation of $(\mathscr{S}, \pi_o)$ as given by Definition 20 is a reverse interval update of $(F_{\mathscr{S}}, D_{\mathscr{S}}, \mathbb{R}^{n_o})$. If the reverse update of o_k is inclusion monotonic for each $k = n_i + 1, \ldots, n_f$ then the reverse interval propagation of $(\mathscr{S}, \pi_o)$ is inclusion monotonic.*

Proof Finite induction yields immediately that the second inclusion in (2) holds.

Let $\boldsymbol{r} \in {}^*\mathbb{IR}^{n_o}$ and $\boldsymbol{x} \in \mathfrak{D}_{\mathscr{S}}$. If there does not exist a $x \in \boldsymbol{x}$ such that $F_{\mathscr{S}}(x) \in \boldsymbol{r}$, then the first inclusion in (2) holds trivially.

Let $x \in \boldsymbol{x}$ such that $F_{\mathscr{S}}(x) \in \boldsymbol{r}$. Then, there exists a sequence of factor values $\{v_k(x)\}_{k=1}^{n_f}$ with $v_1(x) = x_1, \ldots, v_{n_i}(x) = x_{n_i}$ and $\pi_o(v_1(x), \ldots, v_{n_f}(x)) \in \boldsymbol{r}$. Also, since $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{n_f}$ are inclusion functions (see Theorem 2), $(v_1(x), \ldots, v_{n_f}(x)) \in (\boldsymbol{v}_1(\boldsymbol{x}), \ldots, \boldsymbol{v}_{n_f}(\boldsymbol{x}))$ so that $(v_1(x), \ldots, v_{n_f}(x)) \in (\tilde{\boldsymbol{v}}_1, \ldots, \tilde{\boldsymbol{v}}_{n_f})$ prior to entering the loop. In the following, let $\tilde{\boldsymbol{v}}_k^l$ denote the value of $\tilde{\boldsymbol{v}}_k$ for the given $\boldsymbol{x}$ and $\boldsymbol{r}$ after the lth reverse update, $l = 1, \ldots, n_f - n_i$. Since $o_{n_f}^{\text{rev}}$ is a reverse interval update, it follows that $(v_1(x), \ldots, v_{n_f-1}(x)) \in (\tilde{\boldsymbol{v}}_1^1, \ldots, \tilde{\boldsymbol{v}}_{n_f-1}^1)$. Finite induction yields that $(v_1(x), \ldots, v_{n_i}(x)) \in (\tilde{\boldsymbol{v}}_1^{n_f-n_i}, \ldots, \tilde{\boldsymbol{v}}_{n_i}^{n_f-n_i}) \equiv \boldsymbol{F}_{\mathscr{S}}^{\text{rev}}(\boldsymbol{x}, \boldsymbol{r})$. Thus, $x \in \boldsymbol{F}_{\mathscr{S}}^{\text{rev}}(\boldsymbol{x}, \boldsymbol{r})$ and the first inclusion in (2) holds.

Assume now that o_k^{rev} is inclusion monotonic for each $k = n_i + 1, \ldots, n_f$. Let $\boldsymbol{x}^1, \boldsymbol{x}^2 \in \mathfrak{D}_{\mathscr{S}}$ with $\boldsymbol{x}^1 \subset \boldsymbol{x}^2$ and $\boldsymbol{r}^1, \boldsymbol{r}^2 \in {}^*\mathbb{IR}^{n_o}$ with $\boldsymbol{r}^1 \subset \boldsymbol{r}^2$. Then, $(\tilde{\boldsymbol{v}}_1(\boldsymbol{x}^1, \boldsymbol{r}^1), \ldots, \tilde{\boldsymbol{v}}_{n_f}(\boldsymbol{x}^1, \boldsymbol{r}^1)) \subset (\tilde{\boldsymbol{v}}_1(\boldsymbol{x}^2, \boldsymbol{r}^2), \ldots, \tilde{\boldsymbol{v}}_{n_f}(\boldsymbol{x}^2, \boldsymbol{r}^2))$ prior to entering the loop. Since $o_{n_f}^{\text{rev}}$ is inclusion monotonic,

$(\tilde{\boldsymbol{v}}_1^1(\boldsymbol{x}^1, \boldsymbol{r}^1), \ldots, \tilde{\boldsymbol{v}}_{n_f}^1(\boldsymbol{x}^1, \boldsymbol{r}^1)) \subset (\tilde{\boldsymbol{v}}_1^1(\boldsymbol{x}^2, \boldsymbol{r}^2), \ldots, \tilde{\boldsymbol{v}}_{n_f}^1(\boldsymbol{x}^2, \boldsymbol{r}^2))$. Using finite induction over l yields that $(\tilde{\boldsymbol{v}}_1^{n_f-n_i}(\boldsymbol{x}^1, \boldsymbol{r}^1), \ldots, \tilde{\boldsymbol{v}}_{n_i}^{n_f-n_i}(\boldsymbol{x}^1, \boldsymbol{r}^1)) \subset (\tilde{\boldsymbol{v}}_1^{n_f-n_i}(\boldsymbol{x}^2, \boldsymbol{r}^2), \ldots, \tilde{\boldsymbol{v}}_{n_i}^{n_f-n_i}(\boldsymbol{x}^2, \boldsymbol{r}^2))$. Thus, it follows that $\boldsymbol{F}_{\mathscr{S}}^{\text{rev}}(\boldsymbol{x}^1, \boldsymbol{r}^1) \subset \boldsymbol{F}_{\mathscr{S}}^{\text{rev}}(\boldsymbol{x}^2, \boldsymbol{r}^2)$. □

Next, we will present a result very closely related to Theorem 5 that relies more on standard concepts from interval analysis.

Theorem 6 *Consider $(\mathscr{S}, \pi_o)$ and assume that for each $k = n_i + 1, \ldots, n_f$, the reverse interval update of o_k is inclusion monotonic. Define $F_{\mathscr{S}}^{\text{rev}} : D \times \mathbb{R}^{n_o} \to \mathbb{R}_{\emptyset}^{n_i}$ for each $x \in D$ and $r \in \mathbb{R}^{n_o}$ by*

$$F_{\mathscr{S}}^{\text{rev}}(x, r) = \begin{cases} x & \text{if } F_{\mathscr{S}}(x) = r, \\ \text{NaN} & \text{otherwise.} \end{cases} \tag{3}$$

Then, $\boldsymbol{F}_{\mathscr{S}}^{\text{rev}}$ is an inclusion function of $F_{\mathscr{S}}^{\text{rev}}$ on $\mathfrak{D}_{\mathscr{S}} \times {}^*\mathbb{IR}^{n_o}$.

Proof Let $r \in {}^*\mathbb{R}^{n_o}$. First, consider $x \in D$ so that $F_{\mathscr{S}}(x) = r$. Since $\boldsymbol{F}_{\mathscr{S}}$ is an interval extension of $F_{\mathscr{S}}$, each factor is a degenerate interval after the forward evaluation with $\boldsymbol{v}_k([x, x]) = [v_k(x), v_k(x)]$. Since $F_{\mathscr{S}}(x) = r$, the intersections during the reverse interval propagation return degenerate intervals so that it is clear that $\boldsymbol{F}_{\mathscr{S}}^{\text{rev}}$ is an interval extension of $F_{\mathscr{S}}^{\text{rev}}$ for such $[x, x]$. If $x \in D$ such that $F_{\mathscr{S}}(x) \neq r$ then $\pi_o(\tilde{\boldsymbol{v}}_1, \ldots, \tilde{\boldsymbol{v}}_{n_f}) := \pi_o(\tilde{\boldsymbol{v}}_1, \ldots, \tilde{\boldsymbol{v}}_{n_f}) \cap \boldsymbol{r}$ results in $\tilde{\boldsymbol{v}}_k = \emptyset$ for at least one $k \in \{1, \ldots, n_f\}$. For each $k \in \{1, \ldots, n_f\}$, $\tilde{\boldsymbol{v}}_k$ influences at least one $\tilde{\boldsymbol{v}}_j$ with $j \in \{1, \ldots, n_i\}$ through a sequence of reverse interval updates. Any reverse interval update involving empty intervals yields empty intervals because it is an interval operation. Hence, once the loop is executed, $\tilde{\boldsymbol{v}}_1 \times \cdots \times \tilde{\boldsymbol{v}}_{n_i} = \emptyset$ so that $\boldsymbol{F}_{\mathscr{S}}^{\text{rev}}([x, x], [r, r]) = \emptyset = [\text{NaN}, \text{NaN}] = [F_{\mathscr{S}}^{\text{rev}}(x, r), F_{\mathscr{S}}^{\text{rev}}(x, r)]$. Thus, $\boldsymbol{F}_{\mathscr{S}}^{\text{rev}}$ is an interval extension of $F_{\mathscr{S}}^{\text{rev}}$. Inclusion monotonicity of $\boldsymbol{F}_{\mathscr{S}}^{\text{rev}}$ has been established in Theorem 5. The assertion follows then from Theorem 1. □

Here, we will demonstrate how to obtain inclusion monotonic reverse interval updates for the case of addition. Similar constructions are possible for multiplication and univariate operations [62].

Lemma 1 *Consider $(+, \mathbb{R}^2, \mathbb{R})$. The function $(+^{\text{rev}}, \mathbb{I}_{\emptyset}\mathbb{R}^2 \times {}^*\mathbb{IR}, \mathbb{I}_{\emptyset}\mathbb{R}^2)$ defined for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{I}_{\emptyset}\mathbb{R}$ and $\boldsymbol{r} \in {}^*\mathbb{IR}$ by*

$$+^{\text{rev}}((\boldsymbol{x}, \boldsymbol{y}), \boldsymbol{r}) = (\boldsymbol{r} - \boldsymbol{y}, \boldsymbol{r} - \boldsymbol{x}) \cap (\boldsymbol{x}, \boldsymbol{y})$$

is an inclusion monotonic reverse interval update of $(+, \mathbb{R}^2, \mathbb{R})$.

5 Reverse McCormick propagation

In this section, the ideas for reverse interval propagation are extended to McCormick objects. Again, the enclosure property will be established, but also coherent concavity and inclusion monotonicity of the resulting relaxations will be proved.

Definition 21 Suppose $F : D \subset \mathbb{R}^n \to \mathbb{R}^m$. Consider $\mathscr{F}^{\text{rev}} : \mathbb{M}_{\emptyset}D \times {}^*\mathbb{MR}^m \to \mathbb{M}_{\emptyset}\mathbb{R}^n$. If for all $\mathscr{X} \in \mathbb{M}_{\emptyset}D$ and $\mathscr{R} \in {}^*\mathbb{MR}^m$ it holds that

$$\{x \in \text{Enc}(\mathscr{X}) : F(x) \in \text{Enc}(\mathscr{R})\} \subset \{x \in \text{Enc}(\mathscr{F}^{\text{rev}}(\mathscr{X}, \mathscr{R}))\} \tag{4}$$

and $\mathscr{F}^{\text{rev}}(\mathscr{X}, \mathscr{R}) \subset \mathscr{X}$, then $\mathscr{F}^{\text{rev}}$ is called a *reverse McCormick update* of F.

Definition 22 Let $(\mathscr{S}, \pi_o)$ be a $\mathscr{L}$-computational sequence with n_i inputs and n_o outputs with natural McCormick extension $(\mathscr{F}_{\mathscr{S}}, \mathscr{D}_{\mathscr{S}}, \mathbb{R}^{n_o})$. Let $\mathscr{X} \in \mathscr{D}_{\mathscr{S}}$. Suppose $\mathscr{V}_1(\mathscr{X}), \ldots, \mathscr{V}_{n_f}(\mathscr{X})$ have been calculated according to Definition 18. Let $o_k^{\text{rev}} : \mathbb{M}_{\emptyset} B_k \times {}^*\mathbb{MR} \to \mathbb{M}_{\emptyset} B_k$ be a reverse McCormick update of o_k for each $k = n_i + 1, \ldots, n_f$. Suppose $\tilde{\mathscr{V}}_1, \ldots, \tilde{\mathscr{V}}_{n_f} \in \mathbb{M}_{\emptyset}\mathbb{R}$ are calculated for any $\mathscr{X} \in \mathscr{D}_{\mathscr{S}}$ and $\mathscr{R} \in {}^*\mathbb{IR}^{n_o}$ by the following procedure:

```
$(\tilde{\mathscr{V}}_1, \ldots, \tilde{\mathscr{V}}_{n_f}) := (\mathscr{V}_1(\mathscr{X}), \ldots, \mathscr{V}_{n_f}(\mathscr{X}))$
$\pi_o(\tilde{\mathscr{V}}_1, \ldots, \tilde{\mathscr{V}}_{n_f}) := \pi_o(\tilde{\mathscr{V}}_1, \ldots, \tilde{\mathscr{V}}_{n_f}) \cap \mathscr{R}$
for $l := 1, \ldots, n_f - n_i$ do
    $\pi_{n_f-l+1}(\tilde{\mathscr{V}}_1, \ldots, \tilde{\mathscr{V}}_{n_f-l}) := o^{\text{rev}}_{n_f-l+1}(\pi_{n_f-l+1}(\tilde{\mathscr{V}}_1, \ldots, \tilde{\mathscr{V}}_{n_f-l}), \tilde{\mathscr{V}}_{n_f-l+1})$
end
```

The *reverse McCormick propagation* of $(\mathscr{S}, \pi_o)$ is the function $(\mathscr{F}^{\text{rev}}_{\mathscr{S}}, \mathscr{D}_{\mathscr{S}} \times {}^*\mathbb{MR}^{n_o}, \mathbb{M}_{\emptyset} D_{\mathscr{S}})$ defined for any $\mathscr{X} \in \mathscr{D}_{\mathscr{S}}$ and $\mathscr{R} \in {}^*\mathbb{MR}^{n_o}$ by $\mathscr{F}^{\text{rev}}_{\mathscr{S}}(\mathscr{X}, \mathscr{R}) \equiv \tilde{\mathscr{V}}_1 \times \cdots \times \tilde{\mathscr{V}}_{n_i}$.

Theorem 7 *The reverse McCormick propagation of $(\mathscr{S}, \pi_o)$ as given by Definition 22 is a reverse McCormick update of $(F_{\mathscr{S}}, D_{\mathscr{S}}, \mathbb{R}^{n_o})$.*

Proof Let $\mathscr{R} \in {}^*\mathbb{MR}^m$ and $\mathscr{X} \in \mathscr{D}_{\mathscr{S}}$. Finite induction yields immediately that $\mathscr{F}^{\text{rev}}(\mathscr{X}, \mathscr{R}) \subset \mathscr{X}$. If there does not exist $x \in \text{Enc}(\mathscr{X})$ such that $F_{\mathscr{S}}(x) \in \text{Enc}(\mathscr{R})$, then (4) holds trivially.

Let $x \in \text{Enc}(\mathscr{X})$ satisfy $F_{\mathscr{S}}(x) \in \text{Enc}(\mathscr{R})$. Then, there exists a sequence of factor values $\{v_k(x)\}_{k=1}^{n_f}$ with $v_1(x) = x_1, \ldots, v_{n_i}(x) = x_{n_i}$ and $\pi_o(v_1(x), \ldots, v_{n_f}(x)) \in \text{Enc}(\mathscr{R})$. Also, since $\mathscr{V}_1, \ldots, \mathscr{V}_{n_f}$ are relaxation functions, $(v_1(x), \ldots, v_{n_f}(x)) \in \text{Enc}((\mathscr{V}_1(\mathscr{X}), \ldots, \mathscr{V}_{n_f}(\mathscr{X})))$ so that $(v_1(x), \ldots, v_{n_f}(x)) \in \text{Enc}((\tilde{\mathscr{V}}_1, \ldots, \tilde{\mathscr{V}}_{n_f}))$ prior to entering the loop.

In the following, let $\tilde{\mathscr{V}}_k^l$ denote the value of $\tilde{\mathscr{V}}_k$ for the given $\mathscr{X}$ and $\mathscr{R}$ after the lth reverse update, $l = 1, \ldots, n_f - n_i$. Since $o^{\text{rev}}_{n_f}$ is a reverse McCormick update, it follows that $(v_1(x), \ldots, v_{n_f-1}(x)) \in \text{Enc}(\tilde{\mathscr{V}}_1^1, \ldots, \tilde{\mathscr{V}}^1_{n_f-1})$. Finite induction yields that $(v_1(x), \ldots, v_{n_i}(x)) \in \text{Enc}((\tilde{\mathscr{V}}_1^{n_f-n_i}, \ldots, \tilde{\mathscr{V}}_{n_i}^{n_f-n_i})) \equiv \text{Enc}(\mathscr{F}^{\text{rev}}_{\mathscr{S}}(\mathscr{X}, \mathscr{R}))$. Thus, $x \in \text{Enc}(\mathscr{F}^{\text{rev}}_{\mathscr{S}}(\mathscr{X}, \mathscr{R}))$ and (4) holds. □

Lemma 2 *Consider $(\mathscr{S}, \pi_o)$ and assume that for each $k = n_i + 1, \ldots, n_f$, the reverse McCormick update of o_k is coherently concave and inclusion monotonic on $\mathbb{M}_{\emptyset} B_k \times {}^*\mathbb{MR}$. Then, $\mathscr{F}^{\text{rev}}_{\mathscr{S}}$ is coherently concave and inclusion monotonic on $\mathscr{D}_{\mathscr{S}} \times {}^*\mathbb{MR}^{n_o}$.*

Proof Compositions of coherently concave and inclusion monotonic functions are coherently concave and inclusion monotonic [51, Lemma 2.4.15]. The result thus follows from finite induction, analogous to the proof of Theorem 7. □

Theorem 8 *Consider $(\mathscr{S}, \pi_o)$ and assume that for each $k = n_i + 1, \ldots, n_f$, the reverse McCormick update of o_k is coherently concave and inclusion monotonic. Then, $\mathscr{F}^{\text{rev}}_{\mathscr{S}}$ is a relaxation function of $F^{\text{rev}}_{\mathscr{S}}$ on $\mathscr{D}_{\mathscr{S}} \times {}^*\mathbb{MR}^{n_o}$.*

Proof Let $r \in {}^*\mathbb{R}^{n_o}$. First, consider $x \in D$ so that $F_{\mathscr{S}}(x) = r$. It is clear that $\mathscr{F}^{\text{rev}}_{\mathscr{S}}$ is a McCormick extension of $F^{\text{rev}}_{\mathscr{S}}$ for such $([x, x], [x, x])$ since $\mathscr{F}_{\mathscr{S}}$ is a McCormick extension of $F_{\mathscr{S}}$ and $o_k^{\text{rev}}(\mathscr{B}, \mathscr{R}) \subset \mathscr{B}$ for all $(\mathscr{B}, \mathscr{R}) \in \mathbb{M}_{\emptyset} B_k \times {}^*\mathbb{MR}$ by definition. If $x \in D$ such that $F_{\mathscr{S}}(x) \neq r$ then $\pi_o(\tilde{\mathscr{V}}_1, \ldots, \tilde{\mathscr{V}}_{n_f}) := \pi_o(\tilde{\mathscr{V}}_1, \ldots, \tilde{\mathscr{V}}_{n_f}) \cap ([r, r], [r, r])$ results in $\tilde{\mathscr{V}}_k = \emptyset$ for at least one $k \in \{1, \ldots, n_f\}$. For each $k \in \{1, \ldots, n_f\}$, $\tilde{\mathscr{V}}_k$ influences at least one $\tilde{\mathscr{V}}_j$ with

$j \in \{1, \ldots, n_i\}$ through a sequence of reverse McCormick updates. Any reverse McCormick update involving empty McCormick objects yields empty McCormick objects because it is a McCormick operation. Hence, once the loop is executed, $\tilde{\mathscr{V}}_1 \times \cdots \times \tilde{\mathscr{V}}_{n_i} = \emptyset$ so that $\mathscr{F}^{\mathrm{rev}}_{\mathscr{S}}(([x,x],[x,x]),([r,r],[r,r])) = \emptyset$. Thus, $\mathscr{F}^{\mathrm{rev}}_{\mathscr{S}}$ is a McCormick extension of $F^{\mathrm{rev}}_{\mathscr{S}}$. The assertion follows from Lemma 2 in conjunction with Theorem 3. □

5.1 Reverse McCormick updates of binary operations

Lemma 3 *Consider* $(+, \mathbb{R}^2, \mathbb{R})$ *and its relaxation function* $(+, \mathbb{MR}^2, \mathbb{MR})$. *The function* $(+^{\mathrm{rev}}, \mathbb{M}_{\emptyset}\mathbb{R}^2 \times {}^*\mathbb{MR}, \mathbb{M}_{\emptyset}\mathbb{R}^2)$ *defined for all* $\mathscr{X}, \mathscr{Y} \in \mathbb{M}_{\emptyset}\mathbb{R}$ *and* $\mathscr{R} \in {}^*\mathbb{MR}$ *by*

$$+^{\mathrm{rev}}((\mathscr{X},\mathscr{Y}),\mathscr{R}) = (\mathscr{R}-\mathscr{Y}, \mathscr{R}-\mathscr{X}) \cap (\mathscr{X},\mathscr{Y})$$

is a reverse McCormick update of $(+, \mathbb{R}^2, \mathbb{R})$.

Proof Let $\mathscr{X}, \mathscr{Y} \in \mathbb{M}_{\emptyset}\mathbb{R}$ and $\mathscr{R} \in {}^*\mathbb{MR}$. If $\mathrm{Enc}(+(\mathscr{X},\mathscr{Y}) \cap \mathscr{R}) = \emptyset$, then $\nexists (x,y,r) \in \mathrm{Enc}((\mathscr{X},\mathscr{Y},\mathscr{R})) : r - y = x$. Thus, $r - y \notin \mathrm{Enc}(\mathscr{X})$ for all $(y,r) \in \mathrm{Enc}((\mathscr{Y},\mathscr{R}))$ so that $\mathrm{Enc}(\mathscr{R}-\mathscr{Y}) \cap \mathrm{Enc}(\mathscr{X}) = \emptyset$. Similarly, $\mathrm{Enc}(\mathscr{R}-\mathscr{X}) \cap \mathrm{Enc}(\mathscr{Y}) = \emptyset$ so that (4) holds trivially.

Otherwise, pick $(x,y) \in \mathrm{Enc}(\mathscr{X}) \times \mathrm{Enc}(\mathscr{Y})$ so that $x+y \in \mathrm{Enc}(\mathscr{R})$. Since $\check{\phi} \le x+y \le \hat{\phi}$ and $(x,y) \in ([\check{x},\hat{x}],[\check{y},\hat{y}])$, it follows that $x \ge \check{\phi} - y \ge \check{\phi} - \hat{y}$ and $x \le \hat{\phi} - y \le \hat{\phi} - \check{y}$ and that $y \ge \check{\phi} - x \ge \check{\phi} - \hat{x}$ and $y \le \hat{\phi} - x \le \hat{\phi} - \check{x}$ so that $(x,y) \in \mathrm{Enc}(+^{\mathrm{rev}}((\mathscr{X},\mathscr{Y}),\mathscr{R}))$. Thus, (4) holds. □

Let $(\Gamma, \mathbb{I}_{\emptyset}\mathbb{R} \times {}^*\mathbb{IR} \times \mathbb{I}_{\emptyset}\mathbb{R}, \mathbb{I}_{\emptyset}\mathbb{R})$ denote the *Gauss–Seidel operator* for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{I}_{\emptyset}\mathbb{R}$ and $\boldsymbol{r} \in {}^*\mathbb{IR}$, see [43, Proposition 4.2.1] for its description.

Definition 23 Define an *extension of the Gauss–Seidel operator* to $\mathbb{MR}$, denoted as $\mathscr{G}$: $\mathbb{M}_{\emptyset}\mathbb{R} \times {}^*\mathbb{MR} \times \mathbb{M}_{\emptyset}\mathbb{R} \to \mathbb{M}_{\emptyset}\mathbb{R}$, for all $\mathscr{X}, \mathscr{Y} \in \mathbb{M}_{\emptyset}\mathbb{R}$ and $\mathscr{R} \in {}^*\mathbb{MR}$ by $(\mathscr{G}(\mathscr{X},\mathscr{R},\mathscr{Y}))^B = \Gamma(\boldsymbol{x}^B, \boldsymbol{r}^B, \boldsymbol{y}^B)$ and

$$(\mathscr{G}(\mathscr{X},\mathscr{R},\mathscr{Y}))^C = \begin{cases} (\mathscr{R}' \times \frac{1}{\mathscr{X}'})^C \cap (\boldsymbol{y}')^C & \text{if } 0 \notin \boldsymbol{x}^B, \\ \Gamma(\boldsymbol{x}^B, \boldsymbol{r}^B, \boldsymbol{y}^B) \cap (\boldsymbol{y}')^C & \text{if } 0 \in \boldsymbol{x}^B, 0 \notin \boldsymbol{r}^B, \\ (\boldsymbol{y}')^C & \text{otherwise,} \end{cases}$$

where $\mathscr{X}' = (\boldsymbol{x}^B, \boldsymbol{x}^B \cap \boldsymbol{x}^C)$, $\mathscr{Y}' = (\boldsymbol{y}^B, \boldsymbol{y}^B \cap \boldsymbol{y}^C)$ and $\mathscr{R}' = (\boldsymbol{r}^B, \boldsymbol{r}^B \cap \boldsymbol{r}^C)$.

Lemma 4 *Suppose* $\mathscr{X}, \mathscr{Y} \in \mathbb{M}_{\emptyset}\mathbb{R}$, $\mathscr{R} \in {}^*\mathbb{MR}$. *Then,* $\mathscr{G}(\mathscr{X},\mathscr{R},\mathscr{Y}) \subset \mathscr{B}$ *and*

$$\mathrm{Enc}(\mathscr{G}(\mathscr{X},\mathscr{R},\mathscr{Y})) \supset \{y \in \mathrm{Enc}(\mathscr{Y}) : \exists x \in \mathrm{Enc}(\mathscr{X}), r \in \mathrm{Enc}(\mathscr{R}) : xy = r\}. \quad (5)$$

Proof $\Gamma(\boldsymbol{x}^B, \boldsymbol{r}^B, \boldsymbol{b}^B) \subset \boldsymbol{b}^B$ follows from [43, 4.3.2] and it is also clear that $(\mathscr{G}(\mathscr{X},\mathscr{R},\mathscr{Y}))^C \subset (\boldsymbol{y}')^C$, hence $\mathscr{G}(\mathscr{X},\mathscr{R},\mathscr{Y}) \subset \mathscr{Y}' \subset \mathscr{Y}$. It has already been established [43, Proposition 4.2.1] that

$$\Gamma(\boldsymbol{x}^B, \boldsymbol{r}^B, \boldsymbol{y}^B) = \square\{y \in \boldsymbol{b}^B : \exists a \in \boldsymbol{x}^B, r \in \boldsymbol{r}^B : xy = r\}.$$

Next, note that

$$\square\{y \in \boldsymbol{y}^B : \exists x \in \boldsymbol{x}^B, r \in \boldsymbol{r}^B : xy = r\} \supset \{y \in \mathrm{Enc}(\mathscr{Y}) : \\ \exists x \in \mathrm{Enc}(\mathscr{X}), r \in \mathrm{Enc}(\mathscr{R}) : xy = r\}$$

since $\text{Enc}(\mathscr{X}) \subset \boldsymbol{x}^B$, $\text{Enc}(\mathscr{Y}) \subset \boldsymbol{y}^B$, and $\text{Enc}(\mathscr{R}) \subset \boldsymbol{r}^B$. Therefore, (5) holds for the second and third case. Establishing $(\mathscr{G}(\mathscr{X}, \mathscr{R}, \mathscr{Y}))^C \supset \square\{y \in \text{Enc}(\mathscr{Y}) : \exists x \in \text{Enc}(\mathscr{X}), r \in \text{Enc}(\mathscr{R}) : xy = r\}$ is sufficient to show that (5) holds in the first case.

Suppose that $0 \notin \boldsymbol{x}^B$. Consider $y \in \boldsymbol{y}^C$ such that $\exists x \in (\boldsymbol{x}')^C, r \in (\boldsymbol{r}')^C$ with $xy = r$, noting that $x \neq 0$ by assumption. If such y does not exist then $\{y \in \text{Enc}(\mathscr{Y}) : \exists x \in \text{Enc}(\mathscr{X}), r \in \text{Enc}(\mathscr{R}) : xy = r\} = \emptyset$ and (5) holds trivially. If such y exists, then $y = r \times \frac{1}{x}$. Also, $\frac{1}{\mathscr{X}'}$ exists and $\frac{1}{x} \in \text{Enc}(\frac{1}{\mathscr{X}'})$. Since $r \times \frac{1}{x} \in (\mathscr{R}' \times \frac{1}{\mathscr{X}'})^C$, $(\mathscr{G}(\mathscr{X}, \mathscr{R}, \mathscr{Y}))^C \supset \{y \in \text{Enc}(\mathscr{Y}) : \exists x \in \text{Enc}(\mathscr{X}), r \in \text{Enc}(\mathscr{R}) : xy = r\}$. □

Lemma 5 *Consider* $(\times, \mathbb{R}^2, \mathbb{R})$ *and its relaxation function* $(\times, \mathbb{MR}^2, \mathbb{MR})$*. The function* $(\times^{\text{rev}}, \mathbb{M}_\emptyset\mathbb{R}^2 \times {}^*\mathbb{MR}, \mathbb{M}_\emptyset\mathbb{R}^2)$ *defined for all* $\mathscr{X}, \mathscr{Y} \in \mathbb{M}_\emptyset\mathbb{R}$ *and* $\mathscr{R} \in {}^*\mathbb{MR}$ *by*

$$\times^{\text{rev}}((\mathscr{X}, \mathscr{Y}), \mathscr{R}) = (\mathscr{G}(\mathscr{Y}, \mathscr{R}, \mathscr{X}), \mathscr{G}(\mathscr{G}(\mathscr{Y}, \mathscr{R}, \mathscr{X}), \mathscr{R}, \mathscr{Y}))$$

is a reverse McCormick update of $(\times, \mathbb{R}^2, \mathbb{R})$.

Proof Let $\mathscr{X}, \mathscr{Y} \in \mathbb{M}_\emptyset\mathbb{R}$, $\mathscr{R} \in {}^*\mathbb{MR}$. If $\times(\mathscr{X}, \mathscr{Y}) \cap \mathscr{R} = \emptyset$, there does not exist $x \in \text{Enc}(\mathscr{X})$, $y \in \text{Enc}(\mathscr{Y})$ so that $xy \in \text{Enc}(\mathscr{R})$. Thus, (4) holds trivially. Otherwise, pick $(x, y) \in \text{Enc}(\mathscr{X}) \times \text{Enc}(\mathscr{Y})$ so that $xy \in \text{Enc}(\mathscr{R})$. By Lemma 4, $\text{Enc}(\mathscr{G}(\mathscr{Y}, \mathscr{R}, \mathscr{X})) \supset \{\tilde{x} \in \text{Enc}(\mathscr{X}) : \exists \tilde{y} \in \text{Enc}(\mathscr{Y}), z \in \text{Enc}(\mathscr{R}) : \tilde{x}\tilde{y} = z\}$, and hence $x \in \text{Enc}(\mathscr{G}(\mathscr{Y}, \mathscr{R}, \mathscr{X}))$. Likewise, $\{\tilde{y} \in \text{Enc}(\mathscr{Y}) : \exists \tilde{x} \in \text{Enc}(\mathscr{G}(\mathscr{Y}, \mathscr{R}, \mathscr{X})), z \in \text{Enc}(\mathscr{R}) : \tilde{x}\tilde{y} = z\} \subset \text{Enc}(\mathscr{G}(\mathscr{G}(\mathscr{Y}, \mathscr{R}, \mathscr{X}), \mathscr{R}, \mathscr{Y}))$, hence $y \in \text{Enc}(\mathscr{G}(\mathscr{G}(\mathscr{Y}, \mathscr{R}, \mathscr{X}), \mathscr{R}, \mathscr{Y}))$. Thus, $(x, y) \in \text{Enc}(\times^{\text{rev}}((\mathscr{X}, \mathscr{Y}), \mathscr{R}))$ and (4) holds. □

Note that $\times^{\text{rev}}((\mathscr{X}, \mathscr{Y}), \mathscr{R}) = (\mathscr{G}(\mathscr{G}(\mathscr{X}, \mathscr{R}, \mathscr{Y}), \mathscr{R}, \mathscr{X}), \mathscr{G}(\mathscr{X}, \mathscr{R}, \mathscr{Y}))$ is an alternative reverse McCormick update of $(\times, \mathbb{R}^2, \mathbb{R})$.

5.2 Reverse McCormick updates of univariate functions

Lemma 6 *Let* $B \subset \mathbb{R}$ *and consider an injective continuous function* $(u, B, \mathbb{R}) \in \mathscr{L}$*. Furthermore, assume that* $(u^{-1}, \text{range}(u, B), \mathbb{R}) \in \mathscr{L}$ *where* $\text{range}(u, B)$ *refers to the image of* B *under the real-valued function* u*. The function* $(u^{\text{rev}}, \mathbb{M}_\emptyset B \times {}^*\mathbb{MR}, \mathbb{M}_\emptyset\mathbb{R})$ *defined for all* $\mathscr{X} \in \mathbb{M}_\emptyset B$ *and* $\mathscr{R} \in {}^*\mathbb{MR}$ *by*

$$u^{\text{rev}}(\mathscr{X}, \mathscr{R}) = u^{-1}(\mathscr{T}) \cap \mathscr{X}$$

where $\mathscr{T} = (\boldsymbol{r}^B \cap u(\boldsymbol{x}^B), \text{Enc}(\mathscr{R} \cap u(\mathscr{X})))$ *is a reverse McCormick update of* $(u, B, \mathbb{R})$.

Proof Let $\mathscr{X} \in \mathbb{M}_\emptyset B$. Suppose that $\text{Enc}(\mathscr{T}) = \emptyset$. Since $(u, \mathbb{M}_\emptyset B, \mathbb{MR})$ is a relaxation function, there does not exist an $x \in \text{Enc}(\mathscr{X})$ so that $u(x) \in \text{Enc}(\mathscr{R})$. Otherwise, since $(u, B, \mathbb{R})$ is continuous and injective, it is invertible on $\text{range}(u, B)$ and u^{-1} is continuous [46, Thm. 4.17]. Since $(u^{-1}, \text{range}(u, B), \mathbb{R}) \in \mathscr{L}$, $u(x) \in \text{Enc}(\mathscr{T})$ implies $x = u^{-1}(u(x)) \in \text{Enc}(u^{-1}(\mathscr{T}))$. □

Remark 3 Lemma 6 can be used to define the reverse McCormick update of $-(\cdot)$, $(\cdot)^n$ for odd $n \in \mathbb{N}$, exp, log, $\sqrt{\cdot}$, etc. It is also applicable to $\frac{1}{(\cdot)}$ if B is restricted to either the negative or positive reals.

Lemma 7 *Let* $n \in \mathbb{N}$ *be even. Consider* $(u, \mathbb{R}, \mathbb{R}) \in \mathscr{L}$ *where* $u(x) = x^n$ *and assume that* $(\sqrt[n]{\cdot}, [0, +\infty), \mathbb{R}) \in \mathscr{L}$*. The function* $(u^{\text{rev}}, \mathbb{M}_\emptyset\mathbb{R} \times {}^*\mathbb{MR}, \mathbb{M}_\emptyset\mathbb{R})$ *defined for all* $\mathscr{X} \in \mathbb{M}_\emptyset\mathbb{R}$ *and* $\mathscr{R} \in {}^*\mathbb{MR}$ *by*

$$u^{\mathrm{rev}}(\mathscr{X},\mathscr{R}) = \begin{cases} \emptyset & \textit{if } \boldsymbol{r}^B \cap u(\boldsymbol{x}^B) = \emptyset, \\ \sqrt[n]{\mathscr{T}} \cap \mathscr{X} & \textit{if } \underline{x} \geq 0, \\ -\sqrt[n]{\mathscr{T}} \cap \mathscr{X} & \textit{if } \overline{x} \leq 0, \\ \left(u^{\mathrm{rev}}(\boldsymbol{x}^B,\boldsymbol{t}^B), \left[-\sqrt[n]{\hat{t}}, \sqrt[n]{\hat{t}}\right] \cap u^{\mathrm{rev}}(\boldsymbol{x}^B,\boldsymbol{t}^B)\right) \cap \mathscr{X} & \textit{otherwise}, \end{cases}$$

where $\mathscr{T} = (\boldsymbol{t}^B, \boldsymbol{t}^C) = (\boldsymbol{r}^B \cap u(\boldsymbol{x}^B) \cap [0,+\infty), \mathrm{Enc}(\mathscr{R}) \cap \mathrm{Enc}(u(\mathscr{X})) \cap [0,+\infty))$ *is a reverse McCormick update of* $(u, \mathbb{R}, \mathbb{R})$*, and* $u^{\mathrm{rev}}(\boldsymbol{x}^B, \boldsymbol{t}^B)$ *denotes the reverse interval update for the operation.*

Proof Let $\mathscr{X} \in \mathbb{M}_{\emptyset}\mathbb{R}$. Suppose that $\boldsymbol{r}^B \cap u(\boldsymbol{x}^B) = \emptyset$. Since $(u, \mathbb{I}_{\emptyset}\mathbb{R}, \mathbb{IR})$ is an inclusion function, there does not exist an $x \in \boldsymbol{x}^B$ so that $u(x) \in \boldsymbol{r}^B$.

In the following, assume that $\boldsymbol{r}^B \cap u(\boldsymbol{x}^B) \neq \emptyset$. Note that intersecting $\mathscr{R}$ with the nonnegative half space only ensures that no domain violation occurs. Let $\tilde{x} \in \mathrm{Enc}(\mathscr{X})$ so that $u(\tilde{x}) \in \mathrm{Enc}(\mathscr{R})\}$. If $\underline{x} \geq 0$ then $\tilde{x} \geq 0$. By definition of the relaxation function of $\sqrt[n]{\cdot}$, it follows that $\{x \in \mathbb{R} : x \geq 0 \wedge u(x) \in \mathrm{Enc}(\mathscr{R})\} \subset \mathrm{Enc}(\sqrt[n]{\mathscr{T}})$. Similarly, if $\overline{x} \leq 0$ then $\tilde{x} \leq 0$. Since $u(-\tilde{x}) = u(\tilde{x}) \geq 0$ and $u(-\tilde{x}) \in \mathrm{Enc}(\mathscr{R})$, $u(-\tilde{x}) \in \mathrm{Enc}(\mathscr{T})$ so that $\tilde{x} = -(-\tilde{x}) = -\sqrt[n]{u(-\tilde{x})} \in \mathrm{Enc}(-\sqrt[n]{\mathscr{T}})$. Hence, $\{x \in \mathbb{R} : x \leq 0 \wedge u(x) \in \mathrm{Enc}(\mathscr{R})\} \subset \mathrm{Enc}(-\sqrt[n]{\mathscr{T}})$. Otherwise, if $0 \notin \boldsymbol{x}^B$, it is easy to see that

$$\{x \in \mathbb{R} : u(x) \in \mathrm{Enc}(\mathscr{R})\} = \{x \in \mathbb{R} : \exists y \in \mathrm{Enc}(\mathscr{R}), \boldsymbol{x} = -\sqrt[n]{y} \vee \boldsymbol{x} = \sqrt[n]{y}\} \subset \left[-\sqrt[n]{\hat{t}}, \sqrt[n]{\hat{t}}\right].$$

Intersecting with the reverse interval update does not discard any $\tilde{x}$ for which $u(\tilde{x}) \in \mathrm{Enc}(\mathscr{R})$ holds. □

Note that a similar construction is possible to find the reverse McCormick update of the absolute value function.

5.3 Inclusion monotonicity of the reverse McCormick updates

Next, it will be shown that reverse McCormick updates are inclusion monotonic while the next subsection focuses on establishing coherent concavity. Note that [51, Lemma 2.4.15] will be referenced multiple times hereafter to establish inclusion monotonicity of a finite composition of inclusion monotonic functions. Though coherent concavity was also assumed in that result, it is not necessary in order to establish inclusion monotonicity of a finite composition of inclusion monotonic functions.

First, note that the intersection update is inclusion monotonic.

Lemma 8 *The mapping* $\cap : {}^*\mathbb{MR} \times {}^*\mathbb{MR} \to {}^*\mathbb{MR}$ *defined by* $\cap(\mathscr{X},\mathscr{Y}) = \mathscr{X} \cap \mathscr{Y}$ *for all* $\mathscr{X}, \mathscr{Y} \in {}^*\mathbb{MR}$ *is inclusion monotonic on* ${}^*\mathbb{MR} \times {}^*\mathbb{MR}$.

Proof Let $\mathscr{X}_1, \mathscr{X}_2, \mathscr{Y}_1, \mathscr{Y}_2 \in {}^*\mathbb{MR}$. Then, $\mathscr{X}_1 \subset \mathscr{X}_2$ and $\mathscr{Y}_1 \subset \mathscr{Y}_2$ imply $\mathscr{X}_1 \cap \mathscr{Y}_1 \subset \mathscr{X}_2 \cap \mathscr{Y}_2$. □

Next, the binary operations are considered.

Lemma 9 $(+^{\mathrm{rev}}, \mathbb{M}_{\emptyset}\mathbb{R}^2 \times {}^*\mathbb{MR}, \mathbb{M}_{\emptyset}\mathbb{R})$ *is inclusion monotonic on* $\mathbb{M}_{\emptyset}\mathbb{R}^2 \times {}^*\mathbb{MR}$.

Proof This follows immediately since the negative univariate function [51, Theorem 2.4.29 together with Section 2.8], addition [51, Theorem 2.4.20], the intersection operator (Lemma 8) as well as finite composition of inclusion monotonic mappings [51, cf. Lemma 2.4.15] are inclusion monotonic. □

It is helpful to study the extended Gauss–Seidel operator prior to looking at the reverse update of multiplication.

Lemma 10 *$\mathscr{G}$ is inclusion monotonic on $\mathbb{M}_\emptyset\mathbb{R} \times {}^*\mathbb{MR} \times \mathbb{M}_\emptyset\mathbb{R}$.*

Proof It was already shown that multiplication [51, Theorem 2.4.23] and the reciprocal function [51, Theorem 2.4.29 together with Section 2.8] as well as finite composition [51, Lemma 2.4.15] are inclusion monotonic. Also note that Γ is inclusion monotonic [43, 4.3.2]. Let $(\mathscr{X}_1, \mathscr{R}_1, \mathscr{Y}_1), (\mathscr{X}_2, \mathscr{R}_2, \mathscr{Y}_2) \in \mathbb{M}_\emptyset\mathbb{R} \times {}^*\mathbb{MR} \times \mathbb{M}_\emptyset\mathbb{R}$ so that $(\mathscr{X}_1, \mathscr{R}_1, \mathscr{Y}_1) \subset (\mathscr{X}_2, \mathscr{R}_2, \mathscr{Y}_2)$. If $0 \notin \boldsymbol{x}_2^B$ then $(\mathscr{R}_1' \times \frac{1}{\mathscr{X}_1'})^C \cap (\boldsymbol{y}_1')^C \subset (\mathscr{R}_2' \times \frac{1}{\mathscr{X}_2'})^C \cap (\boldsymbol{y}_2')^C$. Otherwise, if $0 \notin \boldsymbol{x}_1^B$ then $(\mathscr{R}_1' \times \frac{1}{\mathscr{X}_1'})^C \cap (\boldsymbol{y}_1')^C \subset \Gamma(\boldsymbol{x}_1^B, \boldsymbol{r}_1^B, \boldsymbol{y}_1^B) \cap (\boldsymbol{y}_1')^C \subset \Gamma(\boldsymbol{x}_2^B, \boldsymbol{r}_2^B, \boldsymbol{y}_2^B) \cap (\boldsymbol{y}_2')^C$. Otherwise, if $0 \in \boldsymbol{x}_1^B$ and $0 \notin \boldsymbol{r}_2^B$ then $\Gamma(\boldsymbol{x}_1^B, \boldsymbol{r}_1^B, \boldsymbol{y}_1^B) \cap (\boldsymbol{y}_1')^C \subset \Gamma(\boldsymbol{x}_2^B, \boldsymbol{r}_2^B, \boldsymbol{y}_2^B) \cap (\boldsymbol{y}_2')^C$. Otherwise, if $0 \in \boldsymbol{x}_1^B$ and $0 \in \boldsymbol{r}_2^B$ then $\Gamma(\boldsymbol{x}_1^B, \boldsymbol{r}_1^B, \boldsymbol{y}_1^B) \cap (\boldsymbol{y}_1')^C \subset (\boldsymbol{y}_1')^C \subset (\boldsymbol{y}_2')^C$ Thus, $\mathscr{G}$ is inclusion monotonic. □

Lemma 11 $(\times^{\text{rev}}, \mathbb{M}_\emptyset\mathbb{R}^2 \times {}^*\mathbb{MR}, \mathbb{M}_\emptyset\mathbb{R})$ *is inclusion monotonic on* $\mathbb{M}_\emptyset\mathbb{R} \times \mathbb{M}_\emptyset\mathbb{R} \times {}^*\mathbb{MR}$.

Proof Since $\mathscr{G}$ and finite composition [51, Lemma 2.4.15] are inclusion monotonic, the result is immediate. □

Next, the reverse updates of univariate functions are considered.

Lemma 12 *Let $B \subset \mathbb{R}$ and consider an injective continuous function $(u, B, \mathbb{R}) \in \mathscr{L}$. Assume that $(u^{-1}, \text{range}(u, B), \mathbb{R}) \in \mathscr{L}$. Then, u^{rev} as defined in Lemma 6 is inclusion monotonic on $\mathbb{M}_\emptyset B \times {}^*\mathbb{MR}$.*

Proof Since $(u^{-1}, \text{range}(u, B), \mathbb{R}) \in \mathscr{L}$, it follows that u^{-1} is inclusion monotonic [51, Theorem 2.4.29]. □

Lemma 13 *Let $n \in \mathbb{N}$ be even. Consider $(u, \mathbb{R}, \mathbb{R}) \in \mathscr{L}$ where $u(x) = x^n$. Assume that $(\sqrt[n]{\cdot}, [0, +\infty), \mathbb{R}) \in \mathscr{L}$. Suppose that the convex and concave envelopes of $\sqrt[n]{\cdot}$ are used in calculating relaxations. Then, u^{rev} as defined in Lemma 7 is inclusion monotonic on $\mathbb{M}_\emptyset\mathbb{R} \times {}^*\mathbb{MR}$.*

Proof Note that the relaxation function of $\sqrt[n]{\cdot}$ and of the negative operator, the intersection operator and finite composition is inclusion monotonic. Note that $\mathscr{T}$ is inclusion monotonic by construction and so is $u^{\text{rev}}(\boldsymbol{x}^B, \boldsymbol{t}^B)$. Let $(\mathscr{X}_1, \mathscr{R}_1), (\mathscr{X}_2, \mathscr{R}_2) \in \mathbb{M}_\emptyset B \times {}^*\mathbb{MR}$ so that $(\mathscr{X}_1, \mathscr{R}_1) \subset (\mathscr{X}_2, \mathscr{R}_2)$. If $\boldsymbol{r}_1^B \cap u(\boldsymbol{x}_1^B) = \emptyset$ or if $\underline{x}_2 \geq 0$ or $\overline{x}_2 \leq 0$ then $u^{\text{rev}}(\mathscr{X}_1, \mathscr{R}_1) \subset u^{\text{rev}}(\mathscr{X}_2, \mathscr{R}_2)$. Otherwise, suppose $\underline{x}_1 \geq 0$. Then $\sqrt[n]{\mathscr{T}_1} \cap \mathscr{X}_1 \subset (u^{\text{rev}}(\boldsymbol{x}_1^B, \boldsymbol{t}_1^B), [-\sqrt[n]{\hat{t}_1}, \sqrt[n]{\hat{t}_1}] \cap u^{\text{rev}}(\boldsymbol{x}_1^B, \boldsymbol{t}_1^B)) \cap \mathscr{X}_1 \subset (u^{\text{rev}}(\boldsymbol{x}_2^B, \boldsymbol{t}_2^B), [-\sqrt[n]{\hat{t}_2}, \sqrt[n]{\hat{t}_2}] \cap u^{\text{rev}}(\boldsymbol{x}_2^B, \boldsymbol{t}_2^B)) \cap \mathscr{X}_2$. A similar argument applies when $\overline{x}_1 \leq 0$. In any other case, inclusion monotonicity follows directly from the properties referenced above and the monotonicity of $\sqrt[n]{\cdot}$, i.e., $\hat{t}_1 \leq \hat{t}_2$ implies that $[-\sqrt[n]{\hat{t}_1}, \sqrt[n]{\hat{t}_1}] \subset [-\sqrt[n]{\hat{t}_2}, \sqrt[n]{\hat{t}_2}]$. □

5.4 Coherent concavity of the reverse McCormick updates

Next, it will be shown that reverse McCormick updates are coherently concave. Note that if either $\text{Enc}(\mathscr{F}(\mathscr{X}_1)) = \emptyset$ or $\text{Enc}(\mathscr{F}(\mathscr{X}_2)) = \emptyset$, then the subset condition for coherent concavity holds trivially. Thus, in the proofs below, this case is never considered explicitly.

First, note that the intersection update is coherently concave.

Lemma 14 *The mapping* $\cap : {}^*\mathbb{MR} \times {}^*\mathbb{MR} \to {}^*\mathbb{MR}$ *defined by* $\cap(\mathscr{X}, \mathscr{Y}) = \mathscr{X} \cap \mathscr{Y}$ *for all* $\mathscr{X}, \mathscr{Y} \in {}^*\mathbb{MR}$ *is coherently concave on* ${}^*\mathbb{MR} \times {}^*\mathbb{MR}$.

Proof Suppose $\mathscr{X}_1, \mathscr{X}_2 \in {}^*\mathbb{MR}$ and $\mathscr{Y}_1, \mathscr{Y}_2 \in {}^*\mathbb{MR}$ are coherent. Let $\lambda \in [0, 1]$. Since $\boldsymbol{x}_1^B = \boldsymbol{x}_2^B$ and $\boldsymbol{y}_1^B = \boldsymbol{y}_2^B$, it follows that $\boldsymbol{x}_1^B \cap \boldsymbol{y}_1^B = \boldsymbol{x}_2^B \cap \boldsymbol{y}_2^B = (\lambda \boldsymbol{x}_1^B + (1-\lambda)\boldsymbol{x}_2^B) \cap (\lambda \boldsymbol{y}_1^B + (1-\lambda)\boldsymbol{y}_2^B)$. Thus, $\cap(\mathscr{X}_1, \mathscr{Y}_1)$ and $\cap(\mathscr{X}_2, \mathscr{Y}_2)$ are coherent.

We will show that $\cap(\text{Conv}(\lambda, (\mathscr{X}_1, \mathscr{Y}_1), (\mathscr{X}_2, \mathscr{Y}_2))) \supset \text{Conv}(\lambda, \cap(\mathscr{X}_1, \mathscr{Y}_1), \cap(\mathscr{X}_2, \mathscr{Y}_2))$. Let $z_1 \in \boldsymbol{x}_1^C \cap \boldsymbol{y}_1^C$ and $z_2 \in \boldsymbol{x}_2^C \cap \boldsymbol{y}_2^C$. Denote $z = \lambda z_1 + (1-\lambda) z_2$. By construction, $z_1 \in \boldsymbol{x}_1^C$, $z_1 \in \boldsymbol{y}_1^C$ and $z_2 \in \boldsymbol{x}_2^C$, $z_2 \in \boldsymbol{y}_2^C$ so that $z \in \lambda \boldsymbol{x}_1^C + (1-\lambda)\boldsymbol{x}_2^C$ and $z \in \lambda \boldsymbol{y}_1^C + (1-\lambda)\boldsymbol{y}_2^C$. Thus, $z \in (\lambda \boldsymbol{x}_1^C + (1-\lambda)\boldsymbol{x}_2^C) \cap (\lambda \boldsymbol{y}_1^C + (1-\lambda)\boldsymbol{y}_2^C)$ so that $\lambda(\boldsymbol{x}_1^C \cap \boldsymbol{y}_1^C) + (1-\lambda)(\boldsymbol{x}_2^C \cap \boldsymbol{y}_2^C) \subset ((\lambda \boldsymbol{x}_1^C + (1-\lambda)\boldsymbol{x}_2^C) \cap (\lambda \boldsymbol{y}_1^C + (1-\lambda)\boldsymbol{y}_2^C))$. □

In particular, note that the proof indicates that $\cap(\mathscr{X}_1, \mathscr{Y}_1) \neq \emptyset$ and $\cap(\mathscr{X}_2, \mathscr{Y}_2) \neq \emptyset$ imply that $\cap(\text{Conv}(\lambda, (\mathscr{X}_1, \mathscr{Y}_1), (\mathscr{X}_2, \mathscr{Y}_2))) \neq \emptyset$.

Next, the binary operations are considered.

Lemma 15 $(+^{\text{rev}}, \mathbb{M}_\emptyset\mathbb{R}^2 \times {}^*\mathbb{MR}, \mathbb{M}_\emptyset\mathbb{R})$ *is coherently concave on* $\mathbb{M}_\emptyset\mathbb{R}^2 \times {}^*\mathbb{MR}$.

Proof Note that negative univariate function [51, Theorems 2.4.29 and 2.4.30 together with Section 2.8], addition [51, Theorem 2.4.20] and the intersection operator (Lemmas 8 and 14) are inclusion monotonic and coherently concave, $(+^{\text{rev}}, \mathbb{I}_\emptyset\mathbb{R}^2 \times {}^*\mathbb{IR}, \mathbb{I}_\emptyset\mathbb{R})$ is inclusion monotonic and finite composition [51, Lemma 2.4.15] is coherently concave. Thus, the result follows. □

It is helpful to study the extended Gauss–Seidel operator prior to looking at the reverse update of multiplication.

Lemma 16 $\mathscr{G}$ *is coherently concave on* $\mathbb{M}_\emptyset\mathbb{R} \times {}^*\mathbb{MR} \times \mathbb{M}_\emptyset\mathbb{R}$.

Proof Let $\mathscr{X}_1, \mathscr{X}_2 \in \mathbb{M}_\emptyset\mathbb{R}$, $\mathscr{R}_1, \mathscr{R}_2 \in {}^*\mathbb{MR}$ and $\mathscr{Y}_1, \mathscr{Y}_2 \in \mathbb{M}_\emptyset\mathbb{R}$ be coherent. Since $\boldsymbol{x}_1^B = \boldsymbol{x}_2^B$, $\boldsymbol{y}_1^B = \boldsymbol{y}_2^B$ and $\boldsymbol{r}_1^B = \boldsymbol{r}_2^B$, it follows that $\Gamma(\boldsymbol{x}_1^B, \boldsymbol{r}_1^B, \boldsymbol{y}_1^B) = \Gamma(\boldsymbol{x}_2^B, \boldsymbol{r}_2^B, \boldsymbol{y}_2^B)$ so that $\mathscr{G}(\mathscr{X}_1, \mathscr{R}_1, \mathscr{Y}_1)$ and $\mathscr{G}(\mathscr{X}_2, \mathscr{R}_2, \mathscr{Y}_2)$ are coherent.

Suppose $0 \notin \boldsymbol{x}_1^B = \boldsymbol{x}_2^B$. It was already shown that multiplication [51, Theorems 2.4.23 and 2.4.24] and the reciprocal function [51, Theorems 2.4.29 and 2.4.30 together with Section 2.8] are inclusion monotonic and coherently concave and finite compositions of inclusion monotonic, coherently concave functions are coherently concave [51, Lemma 2.4.15]. Also, Lemmas 8 and 14 show that the intersection operation is coherently concave and inclusion monotonic. Thus, $\mathscr{G}$ is coherently concave in this case.

Next, suppose $0 \in \boldsymbol{x}_1^B$ and $\underline{r}_1 = \underline{r}_2 > 0$ or $\overline{r}_1 = \overline{r}_2 < 0$. Pick $\lambda \in [0, 1]$ and let $z_1 \in \Gamma(\boldsymbol{x}_1^B, \boldsymbol{r}_1^B, \boldsymbol{y}_1^B) \cap \boldsymbol{y}_1^C$, $z_2 \in \Gamma(\boldsymbol{x}_2^B, \boldsymbol{r}_2^B, \boldsymbol{y}_2^B) \cap \boldsymbol{y}_2^C$. Consider $z = \lambda z_1 + (1-\lambda) z_2$. Since $z_1 \in \boldsymbol{y}_1^C$ and $z_2 \in \boldsymbol{y}_2^C$, $z \in \lambda \boldsymbol{y}_1^C + (1-\lambda)\boldsymbol{y}_2^C$. Note that $z \in \Gamma(\boldsymbol{x}_1^B, \boldsymbol{r}_1^B, \boldsymbol{y}_1^B) = \Gamma(\boldsymbol{x}_2^B, \boldsymbol{r}_2^B, \boldsymbol{y}_2^B)$. Thus, $z \in \lambda(\Gamma(\boldsymbol{x}_1^B, \boldsymbol{r}_1^B, \boldsymbol{y}_1^B) \cap \boldsymbol{y}_1^C) + (1-\lambda)(\Gamma(\boldsymbol{x}_2^B, \boldsymbol{r}_2^B, \boldsymbol{y}_2^B) \cap \boldsymbol{y}_2^C)$ and $\mathscr{G}$ is coherently concave in this case.

In the last case, coherent concavity is immediate. □

Lemma 17 $(\times^{\text{rev}}, \mathbb{M}_\emptyset\mathbb{R}^2 \times {}^*\mathbb{MR}, \mathbb{M}_\emptyset\mathbb{R})$ *is coherently concave on* $\mathbb{M}_\emptyset\mathbb{R} \times \mathbb{M}_\emptyset\mathbb{R} \times {}^*\mathbb{MR}$.

Proof Since $\mathscr{G}$ is inclusion monotonic and coherently concave and finite compositions of inclusion monotonic, coherently concave functions are coherently concave [51, Lemma 2.4.15], the result is immediate. □

Next, the reverse updates of univariate functions are considered.

Lemma 18 *Let $B \subset \mathbb{R}$ and consider an injective continuous function $(u, B, \mathbb{R}) \in \mathscr{L}$. Assume that $(u^{-1}, \text{range}(u, B), \mathbb{R}) \in \mathscr{L}$. Then, u^{rev} as defined in Lemma 6 is coherently concave on $\mathbb{M}_{\emptyset}B \times {}^*\mathbb{MR}$.*

Proof Let $\mathscr{X}_1, \mathscr{X}_2 \in \mathbb{M}_{\emptyset}\mathbb{R}$ and $\mathscr{R}_1, \mathscr{R}_2 \in {}^*\mathbb{MR}$ be coherent, i.e., $\boldsymbol{x}_1^B = \boldsymbol{x}_2^B$ and $\boldsymbol{r}_1^B = \boldsymbol{r}_2^B$. Note that $u^{\text{rev}}(\mathscr{X}_1, \mathscr{R}_1)$ and $u^{\text{rev}}(\mathscr{X}_2, \mathscr{R}_2)$ are coherent. Since $(u^{-1}, \text{range}(u, B), \mathbb{R}) \in \mathscr{L}$, it follows that u^{-1} is coherently concave [51, Theorem 2.4.30]. □

Lemma 19 *Let $n \in \mathbb{N}$ be even. Consider $(u, \mathbb{R}, \mathbb{R}) \in \mathscr{L}$ where $u(x) = x^n$. Assume that $(\sqrt[n]{\cdot}, [0, +\infty), \mathbb{R}) \in \mathscr{L}$. Then, u^{rev} as defined in Lemma 7 is coherently concave on $\mathbb{M}_{\emptyset}\mathbb{R} \times {}^*\mathbb{MR}$.*

Proof Let $\mathscr{X}_1, \mathscr{X}_2 \in \mathbb{M}_{\emptyset}\mathbb{R}$ and $\mathscr{R}_1, \mathscr{R}_2 \in {}^*\mathbb{MR}$ be coherent, i.e., $\boldsymbol{x}_1^B = \boldsymbol{x}_2^B$ and $\boldsymbol{r}_1^B = \boldsymbol{r}_2^B$. Note that $u^{\text{rev}}(\mathscr{X}_1, \mathscr{R}_1)$ and $u^{\text{rev}}(\mathscr{X}_2, \mathscr{R}_2)$ are coherent.

By assumption, the relaxation function of $\sqrt[n]{\cdot}$ is coherently concave. Likewise, $-\sqrt[n]{\cdot}$ is coherently concave which follows from coherent concavity of the negative operator and the composition theorem [51, Lemma 2.4.15]. It follows that $\sqrt[n]{\cdot}$ and $-\sqrt[n]{\cdot}$ are relaxation functions. As the intersection operator is coherently concave (Lemma 14), coherent concavity for the cases $\underline{x} \geq 0$ and $\overline{x} \leq 0$ follows. Otherwise, we must consider two potential roots. Define $(\tilde{u}^{-1}, [0, +\infty), \mathbb{P}(\mathbb{R}))$ for each $x \in [0, +\infty)$ by $\tilde{u}^{-1}(x) = \{-\sqrt[n]{x}, \sqrt[n]{x}\}$ and note that $u(y) = x$ for each $y \in \tilde{u}^{-1}(x)$ and $x \in \tilde{u}^{-1}(u(x))$. It is easy to see that $-\sqrt[n]{\cdot}$ and $\sqrt[n]{\cdot}$ are the convex and concave envelopes of $\tilde{u}^{-1}$ so that we can use the construction of the relaxation function of $\tilde{u}^{-1}$ in Eq. (1). Furthermore, $t^{\min}(\boldsymbol{t}^B) = t^{\max}(\boldsymbol{t}^B) = \overline{t}$ and $\overline{t} \geq \hat{t}$ in this case so that $\text{mid}(\check{t}, \hat{t}, t^{\min}(\boldsymbol{t}^B)) = \text{mid}(\check{t}, \hat{t}, t^{\max}(\boldsymbol{t}^B)) = \hat{t}$. This is equivalent to the relaxation we obtain by using Eq. (1). It has already been established that Eq. (1) provides for a coherently concave relaxation function [51, Theorem 2.4.30] so that, together with Lemma 14, coherent concavity of the last case follows. □

6 Using reverse McCormick propagation in CSPs and in global optimization

Consider a CSP with variables $y = (y_1, \ldots, y_n)$, domains $\boldsymbol{d} \in \mathbb{IR}^n$ and constraints

$$G(y) \leq 0, \tag{6}$$

$$H(y) = 0, \tag{7}$$

where $G : \boldsymbol{d} \to \mathbb{R}^{n_g}$ and $H : \boldsymbol{d} \to \mathbb{R}^{n_h}$ are $\mathscr{L}$-factorable functions.

Suppose that the variables $y \in \boldsymbol{d}$ can be partitioned into *independent* and *dependent* variables, $p \in \boldsymbol{p} \in \mathbb{IR}^{n-m}$ and $z \in \boldsymbol{x} \in \mathbb{IR}^m$, respectively, where $\boldsymbol{p} \times \boldsymbol{x} = \boldsymbol{d}$. Consider the set-valued map $X : \boldsymbol{p} \to \mathbb{P}(\boldsymbol{x})$ defined by: $p \mapsto \{\xi \in \boldsymbol{x} : G(\xi, p) \leq 0, H(\xi, p) = 0\}$. In words, this mapping returns for each $p \in \boldsymbol{p}$ all points in $\boldsymbol{x}$ that are feasible in the constraints (6) and (7) and thus are solutions of the CSP.

Remark 4 It is *not* assumed that $m = n_h$. The proposed method will work for any choice of m. In particular note that is often not possible to find a closed form for x nor is nonempty $X(p)$ or $X(p)$ a singleton immediate in many cases.

In this section, we will first discuss how reverse McCormick propagation can be applied to utilize equality and inequality constraints. Next, we will compare different full-space and reduced-space relaxations of nonlinear programs and we will conclude with a discussion on how to partition the variables into independent and dependent ones.

6.1 Solving CSPs with equality and inequality constraints

For easier notation, define $C : \boldsymbol{d} \to \mathbb{R}^{n_g+n_h}$ with $c_i(y) = g_i(y)$ for $i = 1, \ldots, n_g$ and $c_{i+n_g}(y) = h_i(y)$ for $i = 1, \ldots, n_h$ and introduce $\mathscr{N} \in {}^*\mathbb{MR}^{n_g+n_h}$ with $\mathscr{N}_i = ((-\infty, 0], (-\infty, 0])$, $i = 1, \ldots, n_g$ and $\mathscr{N}_{i+n_g} = ([0, 0], [0, 0])$, $i = 1, \ldots, n_h$. Let $\mathscr{Y}^0 : \boldsymbol{p} \to \mathbb{MR}^n$ where $\mathscr{Y}_i^0 = (\boldsymbol{x}_i, [\underline{x}_i, \overline{x}_i])$ for $i = 1, \ldots, m$ and $\mathscr{Y}_{i+m}^0 = (\boldsymbol{p}_i, [p_i, p_i])$ for $i = 1, \ldots, n - m$.

$\mathscr{Y}^0$ can be interpreted as an a priori enclosure of the solution set of the CSP when $y_{i+m} = p_i$, $i = 1, \ldots, n - m$. Using the idea of constraint propagation on the DAG of C, several avenues to tighten $\mathscr{Y}^0$ exist. First, it is possible to discard parts of $\boldsymbol{d}$ for which it can be guaranteed that no y exists that satisfies Eqs. (6) and (7). Most easily, this can be achieved by reverse interval propagation [62], which considers the bounds only. Second, reverse McCormick propagation provides a means to improve the original bounds and relaxations to find new bounds and relaxations that are at least as tight as the original relaxations and possibly nonconstant.

Let $(\mathscr{S}, \pi_o)$ be a $\mathscr{L}$-computational sequence corresponding to C. Recall the definition of $C_{\mathscr{S}}^{\text{rev}}$, cf. Eq. (3), and note that for each $p \in \boldsymbol{p}$ and $\xi \in X(p)$ there exists a $\phi \in \text{Enc}(\mathscr{N})$ so that $C_{\mathscr{S}}^{\text{rev}}((\xi, p), \phi) = (\xi, p)$. Consider the reverse McCormick propagation of C:

$$\mathscr{C}_{\mathscr{S}}^{\text{rev}}(((\boldsymbol{x}, \boldsymbol{x}), (\boldsymbol{p}, [p, p])), \mathscr{N}) \equiv ((\tilde{\boldsymbol{x}}, [\check{X}(p), \hat{X}(p)]), (\tilde{\boldsymbol{p}}, [\check{P}(p), \hat{P}(p)])). \quad (8)$$

Note that $\mathscr{C}_{\mathscr{S}}^{\text{rev}}$ is a relaxation function of $C_{\mathscr{S}}^{\text{rev}}$ by Theorem 8. As the following theorem shows, one interpretation of Equation 8 is that it defines $\check{X}, \hat{X} : \boldsymbol{p} \to \mathbb{R}^m$, which are convex and concave relaxations of X on $\boldsymbol{p}$, respectively, (and, less interestingly, $\check{P}, \hat{P} : \boldsymbol{p} \to \mathbb{R}^{n-m}$, which are convex and concave relaxations of the identity function P on $\boldsymbol{p}$).

Theorem 9 *Consider $\mathscr{C}_{\mathscr{S}}^{\text{rev}}$, a relaxation function of $C_{\mathscr{S}}^{\text{rev}}$ on $((\boldsymbol{x}, \boldsymbol{x}), (\boldsymbol{p}, [p, p])) \times {}^*\mathbb{MR}^{n_g+n_h}$. Let $\check{X}, \hat{X} : \boldsymbol{p} \to \mathbb{R}^m$ be as defined by Equation 8. Then, $\check{X}, \hat{X}$ are convex and concave relaxations of X on $\boldsymbol{p}$, respectively.*

Proof Let $x \in \boldsymbol{x}$, $p \in \boldsymbol{p}$ and $\phi \in \text{Enc}(\mathscr{N})$. Note that $C_{\mathscr{S}}^{\text{rev}}((x, p), \phi) = (x, p)$ if $C_{\mathscr{S}}(x, p) = \phi$. Since $\mathscr{C}_{\mathscr{S}}^{\text{rev}}$ is a relaxation function of $C_{\mathscr{S}}^{\text{rev}}$, it follows for such (x, p) that

$$\begin{aligned}\text{Enc}(\mathscr{C}_{\mathscr{S}}^{\text{rev}}(((\boldsymbol{x}, \boldsymbol{x}), (\boldsymbol{p}, [p, p])), \mathscr{N})) &\supset \text{Enc}(\mathscr{C}_{\mathscr{S}}^{\text{rev}}(((\boldsymbol{x}, [x, x]), (\boldsymbol{p}, [p, p])), \mathscr{N})) \\ &\supset ([x, x], [p, p]).\end{aligned}$$

In particular, $\check{x}_i(p) \le \inf\{x_i(p)\} \le \sup\{x_i(p)\} \le \hat{x}_i(p), \forall i = 1, \ldots, m$.

Pick $p_1, p_2 \in \boldsymbol{p}$ and $\lambda \in (0, 1)$. Consider $\mathscr{Y}_1 = ((\boldsymbol{x}, \boldsymbol{x}), (\boldsymbol{p}, [p_1, p_1])) \times \mathscr{N}$ and $\mathscr{Y}_2 = ((\boldsymbol{x}, \boldsymbol{x}), (\boldsymbol{p}, [p_2, p_2])) \times \mathscr{N}$. $\mathscr{C}_{\mathscr{S}}^{\text{rev}}$ is coherently concave on $((\boldsymbol{x}, \boldsymbol{x}), (\boldsymbol{p}, [p, p])) \times {}^*\mathbb{MR}^{n_g+n_h}$, so it follows that

$$\mathscr{C}_{\mathscr{S}}^{\text{rev}}(\text{Conv}(\lambda, \mathscr{Y}_1, \mathscr{Y}_2)) \supset \text{Conv}(\lambda, \mathscr{C}_{\mathscr{S}}^{\text{rev}}(\mathscr{Y}_1), \mathscr{C}_{\mathscr{S}}^{\text{rev}}(\mathscr{Y}_2)),$$

which implies that

$$\begin{aligned}\check{X}(\lambda p_1 + (1 - \lambda)p_2) &\le \lambda \check{X}(p_1) + (1 - \lambda)\check{X}(p_2) \\ \hat{X}(\lambda p_1 + (1 - \lambda)p_2) &\ge \lambda \hat{X}(p_1) + (1 - \lambda)\hat{X}(p_2).\end{aligned}$$

Thus, $\check{X}, \hat{X}$ are convex and concave relaxations of X on $\boldsymbol{p}$, respectively. □

In other words, given $p \in \boldsymbol{p}$ and $\xi \in X(p)$, it holds that $\xi \in [\check{X}(p), \hat{X}(p)]$. Also note that a particular possible outcome of the reverse McCormick propagation is

$$\mathscr{C}_{\mathscr{S}}^{\text{rev}}(((\boldsymbol{x}, \boldsymbol{x}), (\boldsymbol{p}, [p, p])), \mathscr{N}) = ((\tilde{\boldsymbol{x}}, \emptyset), (\tilde{\boldsymbol{p}}, \emptyset))),$$

$$\begin{tikzcd}
((\boldsymbol{x},\boldsymbol{x}),(\boldsymbol{p},[p,p])) \arrow[d, "1"] & \left(\overset{4}{\swarrow}\right) & \begin{array}{c}((\tilde{\boldsymbol{x}},[\underset{\vee}{X}(p),\hat{X}(p)]),(\tilde{\boldsymbol{p}},[\underset{\vee}{P}(p),\hat{P}(p)])) \\ \text{or } ((\tilde{\boldsymbol{x}},\emptyset),(\tilde{\boldsymbol{p}},\emptyset))\end{array} \\
\left.\begin{array}{c}(\boldsymbol{G}(\boldsymbol{x},\boldsymbol{p}),[\underset{\vee}{G}(\boldsymbol{x},p),\hat{G}(\boldsymbol{x},p)]) \\ (\boldsymbol{H}(\boldsymbol{x},\boldsymbol{p}),[\underset{\vee}{H}(\boldsymbol{x},p),\hat{H}(\boldsymbol{x},p)])\end{array}\right\} \arrow[rr, "2"] & & \left\{\begin{array}{c}(\boldsymbol{G}(\boldsymbol{x},\boldsymbol{p}),[\underset{\vee}{G}(\boldsymbol{x},p),\hat{G}(\boldsymbol{x},p)]) \\ \cap\,((-\infty,0],(-\infty,0]) \\ (\boldsymbol{H}(\boldsymbol{x},\boldsymbol{p}),[\underset{\vee}{H}(\boldsymbol{x},p),\hat{H}(\boldsymbol{x},p)]) \\ \cap\,([0,0],[0,0])\end{array}\right. \arrow[u, "3"]
\end{tikzcd}$$

Fig. 2 Principle of forward–reverse McCormick update to construct relaxations of the implicit set-valued mapping $X(\cdot)$: forward evaluation of relaxation functions [52] to obtain a particular kind of relaxations of G and H on $\boldsymbol{p}$ (1), intersection with constraint information (2), and reverse propagation of additional information (3). This procedure can be iterated on if desired (4)

in which case $X(p) = \emptyset$.

The sequence of the calculations for the reverse update $\mathscr{C}_{\mathscr{S}}^{\text{rev}}$ is outlined in Fig. 2. In contrast to the evaluation of natural McCormick extensions, the forward evaluation of the relaxation functions in Step (1) is initialized differently. The results of this evaluation are interval bounds on the range of G and H on $\boldsymbol{x} \times \boldsymbol{p}$, as well as a particular kind of relaxations of G and H on $\boldsymbol{p}$, here denoted by $\underset{\vee}{G}(\boldsymbol{x}, p)$, etc. From the properties of the relaxation function it follows that $\underset{\vee}{G}(\boldsymbol{x}, \cdot)$ is convex on $\boldsymbol{p}$ and that $\underset{\vee}{g}_i(\boldsymbol{x}, p) \le g_i(x, p)$, $\forall(x, p) \in \boldsymbol{x} \times \boldsymbol{p}$ and $i = 1, \ldots, n_g$. Similarly, $\hat{G}(\boldsymbol{x}, p)$ denotes an analogue concave relaxation of G. In Step (2), the constraint information is intersected with the relaxation functions of the constraints. This tightens the relaxations without losing the convexity and concavity properties. Step (3) propagates this information back to the variables so that we obtain relaxations of X evaluated at p or the information that $X(p) = \emptyset$. It is also shown that the procedure can be repeated in order to further improve the computed relaxations (Step (4)).

Let $\mathscr{Y}^{k+1} = \mathscr{C}_{\mathscr{S}}^{\text{rev}}(\mathscr{Y}^k, \mathscr{N})$, $k = 0, 1, \ldots$. Note that the coherent concavity property of $\mathscr{Y}^k$ is guaranteed only for a fixed k so that it is important that the number of reverse updates is equal for all $p \in \boldsymbol{p}$.

Avoiding domain violations Definition 3 ensures that the natural function $F_{\mathscr{S}}$ of a computational sequence $(\mathscr{S}, \pi_o)$, and, in particular, each participating univariate function, is defined at each point of its natural domain $D_{\mathscr{S}}$ and hence can be safely evaluated there. However, the natural domain of a complicated computational sequence is not easily obtained. If the natural function is evaluated at a point outside its domain, which is possible due to difficulty in practically establishing the exact natural domain, the domain of at least one univariate function will be violated. Additionally, Definition 10 further restricts the natural domains of the natural interval and McCormick extensions. Due to the inherent conservatism of the interval and McCormick techniques, domain violations are also potentially possible for $\boldsymbol{x} \in \mathbb{I}D_{\mathscr{S}}$ or $\mathscr{X} \in \mathbb{M}D_{\mathscr{S}}$. In order to avoid either problem, the following convention is implemented. Consider $(u, \boldsymbol{b}, \mathbb{R}) \in \mathscr{L}$ with $\boldsymbol{b} \in \mathbb{IR}$, as is the case for many common univariate functions. If $x \notin \boldsymbol{b}$ then set $u(x) = \text{NaN}$. For $\boldsymbol{x} \in \mathbb{IR}$ or $\mathscr{X} \in \mathbb{MR}$ with $\boldsymbol{x} \not\subset \boldsymbol{b}$ or $\boldsymbol{x}^B \not\subset \boldsymbol{b}$, the evaluation of $u(\boldsymbol{x})$ or $u(\mathscr{X})$ is undefined whereas $u(\boldsymbol{x} \cap \boldsymbol{b})$ or $u(\mathscr{X} \cap (\boldsymbol{b}, \boldsymbol{b}))$ is always defined due to the convention used herein that $u(\emptyset) = \emptyset$. Given any $\boldsymbol{x} \in \mathbb{IR}^{n_i}$ or $\mathscr{X} \in \mathbb{MR}^{n_i}$, this approach continues to construct valid enclosures and relaxations of $\tilde{F} : \mathbb{R}^{n_i} \to \mathbb{R}_{\emptyset}^{n_o}$ defined by

$$\tilde{F}(x) = \begin{cases} F_{\mathscr{S}}(x) & \text{if } x \in D_{\mathscr{S}}, \\ \text{NaN} & \text{otherwise.} \end{cases}$$

Points outside the natural domain evaluate to NaN and, by our convention, NaN is an element of any interval so that any interval-valued or McCormick-valued function satisfies the inclusion property for such x. On the other hand, the natural interval or McCormick extensions bound or relax the natural function at each point that is contained in the natural domain by its usual properties. Overall, this convention allows us to circumvent difficulties with domain violations without losing the inclusion or relaxation function properties. In particular, it provides more directly useful information than throwing a flag indicating that a domain violation occurred.

6.2 Constructing relaxations for reduced-space optimization problems

Consider

$$\begin{aligned} f^* = \min_{z\in\boldsymbol{x},p\in\boldsymbol{p}} \quad & f(z,p) \\ \text{s.t.} \quad & G(z,p) \le 0, \\ & H(z,p) = 0, \end{aligned} \tag{P}$$

where $f : \boldsymbol{x} \times \boldsymbol{p} \to \mathbb{R}$, $G : \boldsymbol{x} \times \boldsymbol{p} \to \mathbb{R}^{n_g}$ and $H : \boldsymbol{x} \times \boldsymbol{p} \to \mathbb{R}^{n_h}$ are $\mathscr{L}$-factorable.

Define the set-valued mapping $\phi : \boldsymbol{p} \to \mathbb{P}(\mathbb{R})$ for each $p \in \boldsymbol{p}$ by $\phi(p) = \{f(z,p) : z \in \boldsymbol{x}, G(z,p) \le 0, H(z,p) = 0\}$. It is obvious that $f^* = \min_{p\in\boldsymbol{p}} \inf \phi(p)$.

Let $\tilde{\boldsymbol{x}}$ and $\tilde{\boldsymbol{p}}$ denote the results of a reverse interval update as outlined above and illustrated in Fig. 2. First, note that $\tilde{\boldsymbol{x}} \times \tilde{\boldsymbol{p}}$ is a superset of the feasible region by construction of the reverse interval update. Recall that the procedure described in the previous section provides valid relaxations of the set-valued mapping X, $\check{X}$ and $\hat{X}$. These can be used to calculate generalized relaxation functions of f. To this extent, let $\mathscr{F}$ denote the natural McCormick extension of f and we will define

$$[\check{\phi}(p), \hat{\phi}(p)] \equiv (\mathscr{F}((\tilde{\boldsymbol{x}}, [\check{X}(p), \hat{X}(p)]), (\tilde{\boldsymbol{p}}, [p,p])))^C.$$

Proposition 1 *Consider*

$$\phi^* = \min_{p\in\tilde{\boldsymbol{p}}} \check{\phi}(p). \tag{R1}$$

Then, (R1) is a convex program and $f^* \ge \phi^*$.

Proof (R1) is a convex program since $\tilde{\boldsymbol{p}}$ is a convex set and $\check{\phi}$ is convex on $\tilde{\boldsymbol{p}}$. $f^* \ge \phi^*$ follows immediately from $\check{\phi}(p) \le \inf \phi(p)$ [51, Theorem 2.7.13]. □

Proposition 2 *Let $\check{f}$, $\check{G}$ and $\check{H}$ denote the standard convex McCormick relaxations of f, G and H, respectively, on $\tilde{\boldsymbol{x}} \times \tilde{\boldsymbol{p}}$ and let $\hat{H}$ denote the standard concave McCormick relaxation of H on $\tilde{\boldsymbol{x}} \times \tilde{\boldsymbol{p}}$. Consider*

$$\begin{aligned} f_1 = \min_{z\in\tilde{\boldsymbol{x}},p\in\tilde{\boldsymbol{p}}} \quad & \check{f}(z,p) \\ \text{s.t.} \quad & \check{G}(z,p) \le 0, \\ & \check{H}(z,p) \le 0 \le \hat{H}(z,p), \\ & \check{X}(p) \le z \le \hat{X}(p), \end{aligned} \tag{R2}$$

Then, $f^* \ge f_1 \ge \phi^*$.

Proof It is clear that (R2) is a relaxation of (P) so that $f^* \ge f_1$. Note that $[\check{f}(z,p), \hat{f}(z,p)] = (\mathscr{F}((\tilde{\boldsymbol{x}}, [z,z]), (\tilde{\boldsymbol{p}}, [p,p])))^C$ holds for the standard McCormick relaxation of f on $\tilde{\boldsymbol{x}} \times \tilde{\boldsymbol{p}}$. Inclusion monotonicity of the natural McCormick extensions implies that for any $p \in \tilde{\boldsymbol{p}}$ and $z \in [\check{X}(p), \hat{X}(p)]$, $\mathscr{F}((\tilde{\boldsymbol{x}}, [z,z]), (\tilde{\boldsymbol{p}}, [p,p])) \subset \mathscr{F}((\tilde{\boldsymbol{x}}, [\check{X}(p), \hat{X}(p)]), (\tilde{\boldsymbol{p}}, [p,p]))$ and thus $\check{f}(z,p) \ge \check{\phi}(p)$ so that $f_1 \ge \phi^*$. □

Remark 5 (R1) and (R2) are valid relaxations of (P). It is known that McCormick relaxations can be nonsmooth functions [41]. Thus, while (R2) is a tighter relaxation of (P), it potentially requires the solution of a convex nonsmooth program with nonlinear nonsmooth constraints. While several methods to solve such programs have been proposed (e.g.,[23,31,37]), and some software is available (e.g., [29,36]), this remains a challenging class of problems to solve robustly. The constraints in (R2) can also be linearized using subgradients [41] to construct an outer-approximation. In this case, the consequence of Proposition 2 is no longer guaranteed to hold. On the other hand, convex nonsmooth programs with box-constraints such as (R1) can be solved more readily using methods such as that provided in [34]. Furthermore, (R1) only requires the solution of a $n-m$-dimensional optimization problem whereas (R2) is n-dimensional.

Remark 6 An alternative method to obtain a relaxation of (P) is the auxiliary variable method which introduces additional variables and constraints for each factor that appears in the DAG [54,56–58]. Its relaxations, prior to linearization, are at least as tight as McCormick relaxations [56, p. 127f] and are differentiable functions. However, the dimension of the resulting nonlinear convex optimization problem is (much) larger. It is typically linearized so that the more robust and more efficient linear programming algorithms can be used. Again, no general comparison of the tightness of different relaxations is possible once the linearization is performed. Also, this approach does not include the constraint $\check{X}(p) \leq z \leq \hat{X}(p)$ in the relaxation so no direct comparison with (R1) and (R2) in terms of tightness is possible.

Suppose it is known that *UBD* is a valid upper bound on the optimal objective function value of (P), e.g., there exists a $(z^\dagger, p^\dagger)$ feasible in (P) with $f(z^\dagger, p^\dagger) = UBD$. Similarly, suppose that *LBD* is a valid lower bound on the optimal objective function value, e.g., there does not exists a $(z^\dagger, p^\dagger)$ feasible in (P) with $f(z^\dagger, p^\dagger) < LBD$. Both cases are very common in the context of a branch-and-bound algorithm. Consider

$$\begin{aligned} f^\ddagger = \min_{z \in \boldsymbol{x}, p \in \boldsymbol{p}} \quad & f(z,p) \\ \text{s.t.} \quad & G(z,p) \leq 0, \\ & H(z,p) = 0, \\ & f(z,p) - UBD \leq 0, \\ & LBD - f(z,p) \leq 0. \end{aligned}$$

It is clear that $f^\ddagger = f^*$ since $(z^\dagger, p^\dagger)$ is feasible in (P). However, we can potentially strengthen the relaxations $\check{X}$, $\hat{X}$ and thus also ϕ^* or f^1 by including $f(z,p) - UBD \leq 0$ and $LBD - f(z,p) \leq 0$ in the reverse propagation outlined in Sect. 6.1.

6.3 Partitioning variables

A discussion on how to partition the variables into $\boldsymbol{x}$ and $\boldsymbol{p}$ concludes this section. We begin by analyzing the two extreme cases: $m = 0$ and $m = n$.

First consider $m = 0$. Here, $\mathscr{Y}$ is initialized using a point, i.e., $\mathscr{Y}_i^0 = (\boldsymbol{p}_i, [p_i, p_i])$ for each $i = 1, \ldots, n$, constructing the tightest relaxations of $C(p)$ after the forward evaluation. However, only two outcomes are possible after the reverse propagation, either $\mathscr{Y}_i^1 = (\tilde{\boldsymbol{p}}_i, [p_i, p_i])$ or $\mathscr{Y}_i^1 = (\tilde{\boldsymbol{p}}_i, \emptyset)$. While the latter case indicates that p violates at least one of the constraints, it is not clear how this information can be exploited numerically. For example, it is not clear how to obtain a hyperplane separating infeasible from potentially feasible points.

Next consider $m = n$. In this case, $\mathscr{Y}$ is initialized using the interval bounds, i.e., $\mathscr{Y}_i^0 = (\boldsymbol{p}_i, \boldsymbol{p}_i)$ for each $i = 1, \ldots, n$. This will yield looser relaxations of C after the forward evaluation and since $\mathscr{Y}$ is constant, we will obtain $\mathscr{Y}^1 = (\tilde{\boldsymbol{p}}, \breve{\boldsymbol{p}})$ after the reverse propagation where $\breve{\boldsymbol{p}} \in \mathbb{I}_{\emptyset} P$ is a *box*. Actually, in this case the reverse McCormick propagation yields the same information as the reverse interval propagation given that the exact image for each univariate function is used as the interval extension and the envelopes are used as the relaxations.

The advantages of the proposed method over interval methods are obtained for partitions between the two extremes listed above. A partitioning with $m = n_h$ such that there exists a unique implicit function $X : \boldsymbol{p} \to \boldsymbol{x}$ with $H(X(p), p) = 0$ for all $p \in \boldsymbol{p}$ is more favorable. In our numerical experience, this partitioning gave results that were better compared to interval reverse propagation. Interval Newton methods can be used to verify the existence and uniqueness of X, see [43, Ch. 5]. Additional inequality constraints can be used to reduce $\boldsymbol{x}$ and $\boldsymbol{p}$ further.

Another effective strategy is to partition the variables such that $m = n_h$, and that the resulting occurrence matrix corresponding to the equality constraint system is structurally nonsingular. Note that in general such a partitioning will not be unique. This approach was used in the majority of the test problems described in Sect. 9. One means of finding such a partition is given by the Dulmage–Mendelsohn decomposition [16]. Dulmage and Mendelsohn showed that any occurrence matrix can be transformed to a block structure consisting of up to three parts: an over-determined part, a fully-determined part and an under-determined part. In the types of problems considered here, the over-determined block should never exist, and by specifying p as the variables in the under-determined block, the equation system will become structurally nonsingular, as desired. An automatic implementation of this algorithm is available in MATLAB [38]. This procedure could also be combined with interval Newton methods to further screen the possible choices for good partitions.

7 Implementation

In this section, an implementation of the reverse interval and McCormick propagation in C++ is presented. First, it is briefly discussed how the DAG of a factorable function can be easily constructed. Next, it is shown how forward and reverse interval and McCormick calculations can be performed on this DAG. Lastly, an outward rounding method for McCormick arithmetic is given, which is necessary for the practical application of reverse McCormick propagation. Consider a factorable function $F : \mathbb{R}^n \to \mathbb{R}^m$. In this section, *independent* and *dependent* variables will refer to y and $F(y)$, respectively. The boost interval library is used for the interval calculations [40] and MC++ provides the necessary routines for McCormick objects [11].

7.1 Algorithm implementation

The first step is the parsing of the factorable function to construct the DAG. In C++ this can be easily achieved using function and operator overloading. The DAG is stored as an array. Each element of the array corresponds to one factor of the factorable function including factors for the assignment of independent variables. Each element stores the operation type as well as pointers to its parent element(s), an interval and a McCormick object (as defined by MC++). Optionally, a constant parameter can be stored, which is used to keep track of, for example, constant exponents or factors. While the first n array elements correspond to the independent variables, pointers to the dependent variables must be stored. Note that after the DAG has been constructed, all remaining operations are performed on this DAG object.

Prior to a forward interval/McCormick pass, the interval/McCormick objects of the independent variables are initialized. During the forward pass each factor is visited in sequence and the factor's interval/McCormick object is updated according to the operation type using the pointers to parents' values. After the forward pass, the interval/McCormick objects of the dependent variables store the values, which could have been alternatively calculated using traditional methods.

Prior to a reverse pass, the interval/McCormick objects of the dependent variables are updated based on the information supplied by the constraints. Then, each factor is visited in reverse order. A reverse interval/McCormick update is performed and the parents' interval/McCormick objects are updated accordingly. After the reverse pass, the independent variables now store the updated interval/McCormick values. If during the reverse pass one of the intervals or McCormick objects of a factor is set to the empty set then the calculation can be aborted and the result of the reverse propagation is the empty set.

Note that MC++ also provides functionality to calculate subgradients of the convex and concave relaxations [41]. This functionality is essential when the relaxations are to be used in convex optimization algorithms. The present implementation also provides routines to update the subgradients during the reverse pass accordingly.

Additionally, the implementation allows the user to provide constraints on the domains of intermediate factors. These can avoid domain violations as outlined at the end of Sect. 6.1 and they are already taken into account during the forward interval or McCormick pass.

Lastly, it is possible to generate code automatically, in any programming language, that implements any combination of the discussed computations. Similar to source code transformation in automatic differentiation [20], the produced code can be executed to efficiently evaluate $\check{X}(\cdot)$ and $\hat{X}(\cdot)$, for example.

7.2 Outward rounding

A major cornerstone of interval arithmetic is the idea that interval extensions of expressions and functions must provide a valid, rigorous enclosure of the range of all real-valued results on the domain of interest. Due to the finite precision of floating-point arithmetic, a rigorous computer implementation of interval arithmetic must use directed rounding to achieve this. In the case of intervals, if a calculation produces an interval $\boldsymbol{x} \in \mathbb{IR}$ whose upper bound and lower bound may not be exactly representable as floating-point numbers, the lower bound must be rounded downward (towards $-\infty$) and the upper bound must be rounded upward (towards $+\infty$) in order for all real numbers $x \in \boldsymbol{x}$ to be validly enclosed [43]. With the rounding performed in this way, the interval result is said to be *outward rounded*.

Analogously, the convex and concave relaxations calculated using reverse McCormick propagation must also provide this rigorous enclosure property. In initial attempts to apply reverse McCormick propagation to large problems, allowing the results of floating-point operations to all be rounded in the default manner (i.e. to the nearest floating-point) caused two types of failures in the procedure. The first occurred when the value of the convex underestimator of the McCormick object exceeded that of its concave overestimator. This behavior was often observed in the backward pass when the result of the calculation should have been an equal convex and concave relaxation, but the value was not exactly representable as a floating-point. The other major issue arose from the incorrect assertion of empty intersections, which most often occurred at the start of the backward pass when the results of the forward pass were intersected with constraint information. Simple fixes, such as adding a small error tolerance to the intersection operation, allowed the algorithm to run without failing, but led to erroneous results.

As a result, it was necessary to implement an outward rounding scheme for McCormick arithmetic. For the operations in McCormick arithmetic that only involve a single floating-point operation to define either the convex or concave relaxation at a point, the outward rounding is easily implemented. These operations include addition or subtraction of two McCormick objects, as well as any operation involving a scalar and a McCormick object (note that taking the negative of a number is an exact operation, and so it is not counted as an second operation in this sense). For example, the outward rounded result of the addition of two McCormick objects can be defined as:

$$\mathscr{X} \updownarrow\!(+)\, \mathscr{Y} := \left(\boldsymbol{x}^B \updownarrow\!(+)\, \boldsymbol{y}^B, \left[\underset{\vee}{x} \downarrow\!(+)\, \underset{\vee}{y}, \hat{x} \uparrow\!(+)\, \hat{y}\right]\right) \tag{9}$$

with $\updownarrow\!(+)$ as shorthand for outward rounded addition, and $\downarrow\!(+)$ and $\uparrow\!(+)$ denoting downward and upward rounded addition, respectively. This can be easily implemented in C++ by calling the function `fesetround` provided by the standard library header `fenv.h` with appropriate argument before each of the operations involving the values of the relaxations, and then allowing the interval operations to be performed with a rigorous interval library. Subtraction of McCormick objects and the operations involving scalars are handled analogously.

For operations such as taking the reciprocal or square root of a McCormick object, as well as for binary multiplication, a different procedure is needed. For instance, the definition for the convex underestimator in the reciprocal operation is as follows:

$$\left(\frac{1}{\mathscr{X}}\right)^{cv} = \begin{cases} \dfrac{1}{\mathrm{mid}(\underset{\vee}{x}, \hat{x}, \overline{x})} & \text{if } \underline{x} > 0 \\ \dfrac{1}{\underline{x}} + \dfrac{1/\overline{x} - 1/\underline{x}}{\overline{x} - \underline{x}}(\mathrm{mid}(\underset{\vee}{x}, \hat{x}, \overline{x}) - \underline{x}) & \text{if } \overline{x} < 0. \end{cases} \tag{10}$$

Here, in the case where $\underline{x} > 0$, the correctly rounded result can be obtained as before by first calling `fesetround(FE_DOWNWARD)` to invoke downward rounding, and then performing the division operation. In the case where $\overline{x} < 0$ however, it is not necessarily true that performing each of the individual calculations with downward rounding will lead to a final result that is less than or equal to the true real-valued result. Instead, it is necessary to use outward rounded interval arithmetic for the individual operations, which will be guaranteed to give a valid result, as in [43, Theorem 1.4.1].

This is implemented in MC++ as follows. First, the double precision variables corresponding to $\underline{x}$, $\overline{x}$, and $\mathrm{mid}(\underset{\vee}{x}, \hat{x}, \overline{x})$ are copied into a rigorous interval type. Then, all calculations are performed using outward rounded interval arithmetic, which will potentially widen the intervals if the results of the individual calculations are not floating-point numbers. Finally, the value of the convex underestimator is set to the value of the lower bound of the final interval result. For the reciprocal, this series of operations can be written out somewhat obtusely as:

$$\left(\frac{1}{\mathscr{X}}\right)^{cv} = \begin{cases} 1 \downarrow\!(\div)\, \mathrm{mid}(\underset{\vee}{x}, \hat{x}, \overline{x}) & \text{if } \underline{x} > 0 \\ \left((1 \updownarrow\!(\div)\, \boldsymbol{x}_L) \updownarrow\!(+)\, (1 \updownarrow\!(\div)\, \boldsymbol{x}_U \updownarrow\!(-)\, 1 \updownarrow\!(\div)\, \boldsymbol{x}_L) \updownarrow\!(\div)\, (\boldsymbol{x}_U \updownarrow\!(-)\, \boldsymbol{x}_L) \updownarrow\!(\times)\, (\boldsymbol{x}_M \updownarrow\!(-)\, \boldsymbol{x}_L)\right)^L & \text{if } \overline{x} < 0. \end{cases} \tag{11}$$

where $\boldsymbol{x}_L$, $\boldsymbol{x}_U$, and $\boldsymbol{x}_M$ are the interval objects corresponding to the lower bound, upper bound, and result of the mid operation, respectively, and the operations adjacent to the up/down arrows represent the corresponding outward rounded interval arithmetic operations. The concave overestimator is defined similarly, and formulas can also be written using the exactly the same approach for outward rounded binary multiplication and other univariate operations such as the square root, exponential, and logarithm. These modified operations were added to the MC++ source code and used to generate the numerical results found in the following sections.

8 Illustrative examples

In this section, we will present illustrative case studies that show how enclosures of the solution sets can be obtained from the reverse McCormick propagation and that these compare favorably to the enclosures computed with reverse interval propagation. In the first case study, the constraints define a unique implicit function on $\boldsymbol{p}$. It is taken from [55] and the results are compared. The second case study compares the feasible region of the relaxed program obtained using reverse McCormick propagation to the feasible region of the standard McCormick relaxation. The third case study focuses on constraints defining a non-unique implicit mapping. The fourth case study shows the effect of a reverse interval propagation pre-processing step when there is no feasible z for some p. The fifth case study shows that relaxations can be sensibly calculated even when there are no feasible z for some p in the interior of $\boldsymbol{p}$. The sixth case study demonstrates how information from inequality constraints can be incorporated. The last case study illustrates how relaxations of the objective function can be significantly improved by incorporating information from the constraints.

We only consider univariate functions from the library $\mathscr{L} = \{(\cdot)^l, \sqrt[l]{\cdot}, \log, \exp\}$, $l \in \mathbb{N}$. However, the method can be applied to any other univariate functions that satisfy Assumptions 1 and 2.

8.1 Equality constraints

Example 1 Let $\boldsymbol{x} = [-0.8, -0.3]$ and $\boldsymbol{p} = [6, 9]$. Consider $h(z, p) = z^2 + zp + 4$ with $(z, p) \in \boldsymbol{x} \times \boldsymbol{p}$. Note that $h(z, p) = 0$ implicitly defines a set-valued mapping $x : \boldsymbol{p} \to \mathbb{P}(\boldsymbol{x}) : p \mapsto \{-\frac{p}{2} + \sqrt{\left(\frac{p}{2}\right)^2 - 4}\}$ so that $h(\xi, p) = 0$ for all $p \in \boldsymbol{p}$ and $\xi \in x(p)$. While Fig. 3a shows the result after one iteration of the reverse McCormick propagation, Fig. 3b depicts the effect of 10 reverse propagation iterations. In both figures the relaxations are compared to those calculated using the method presented in [55]. Note that the calculations for 60 different values of p take a total of 0.0021, 0.0039 and 0.014 s in the case of one reverse propagation, ten reverse propagations and one iteration of the parametric Gauss–Seidel method given in [55] with $\lambda = 0.5$, respectively. Thus, the new method is faster and provides tighter relaxations.

Example 2 Let $\boldsymbol{x} = [-3, 5]$ and $\boldsymbol{p} = [-3, 4]$. Consider $h(z, p) = (\sqrt{p+4} - 3)(\log(p^2 + 1) - z)$ with $(z, p) \in \boldsymbol{x} \times \boldsymbol{p}$. Note that $h(z, p) = 0$ implicitly defines a set-valued mapping $x : \boldsymbol{p} \to \mathbb{P}(\boldsymbol{x}) : p \mapsto \{\log(1 + p^2)\}$ so that $h(\xi, p) = 0$ for all $p \in \boldsymbol{p}$ and $\xi \in x(p)$. The results of a single reverse McCormick propagation are shown in Fig. 4a. Additionally, we also show two different relaxations of the non-convex feasible space $\{(z, p) \in \boldsymbol{x} \times \boldsymbol{p} : h(z, p) = 0)\}$ (shown in asterisks). A common way to relax constraints is the construction of a convex outer-approximation of the feasible space by considering $\{(z, p) \in \boldsymbol{x} \times \boldsymbol{p} : \check{h}(z, p) \leq 0 \leq \hat{h}(z, p)\}$ where $\check{h}, \hat{h}$ are standard McCormick relaxations of h on $\boldsymbol{x} \times \boldsymbol{p}$. This set can be traced by plotting the zero level sets of $\check{h}, \hat{h}$, i.e., $\check{h}(z, p) = 0$ and $\hat{h}(z, p) = 0$. An alternative, tighter outer-approximation can be found by computing $\check{h}, \hat{h}$ on $\tilde{\boldsymbol{x}} \times \tilde{\boldsymbol{p}} = [0, 2.834] \times [-3, 4]$ instead. In Fig. 4b, the same information is shown for smaller original intervals, $\boldsymbol{x} = [-3, 3]$ and $\boldsymbol{p} = [-1, 1]$.

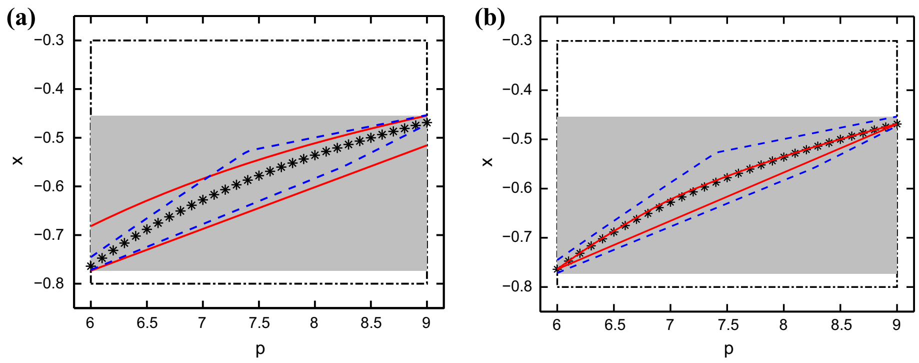

Fig. 3 Result of reverse McCormick propagation for Example 1 showing the original bounds (*dashed-dotted lines*), the improved bounds (*gray box*), the convex and concave relaxations (*solid line*, respectively) as well as results of the set-valued mapping $x(p)$ (asterisks). In **a** one iteration of the reverse McCormick propagation was performed while in **b** the reverse propagation iterations was repeated ten times. The *dashed lines* show convex and concave relaxations calculated using one iteration of the more expensive method in [55]

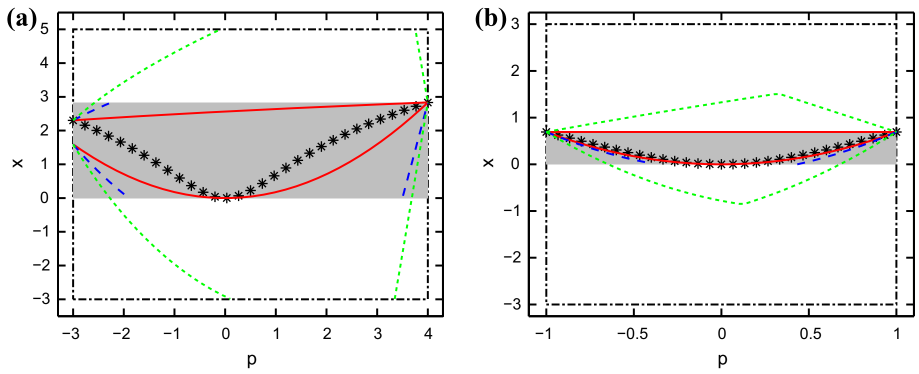

Fig. 4 Result of reverse McCormick propagation for Example 2 showing the original bounds (*dashed-dotted lines*), the improved bounds (*gray box*), the convex and concave relaxations (*solid lines*) as well as results of the set-valued mapping $x(p)$ (*asterisks*). Additionally, zero level sets of the McCormick relaxations of $h(z, p)$ constructed on $\boldsymbol{x} \times \boldsymbol{p}$ (*short dashed lines*) as well as $\tilde{\boldsymbol{x}} \times \tilde{\boldsymbol{p}}$ (*dashed lines*) are shown except where they are outside the interval bounds. Here, the results for different $\boldsymbol{x} \times \boldsymbol{p}$ are shown in **a** and **b**

Example 3 Let $\boldsymbol{x} = [-10, 10]$ and $\boldsymbol{p} = [0, 3]$. Consider $h(z, p) = z^2 - p$ with $(z, p) \in \boldsymbol{x} \times \boldsymbol{p}$. Note that $h(z, p) = 0$ implicitly defines a set-valued mapping $x : \boldsymbol{p} \to \mathbb{P}(\boldsymbol{x}) : p \mapsto \{\sqrt{p}, -\sqrt{p}\}$ so that $h(\xi, p) = 0$ for all $p \in \boldsymbol{p}$ and $\xi \in x(p)$. The results of the reverse McCormick propagation are shown in Fig. 5. Here, no comparison with [55] is possible due to non-uniqueness of x.

Example 4 Let $\boldsymbol{x} = [-10, 10]$ and $\boldsymbol{p} = [0, 3]$. Consider $h(z, p) = z^4 - p^2 + 1$ with $(z, p) \in \boldsymbol{x} \times \boldsymbol{p}$. Note that $h(z, p) = 0$ implicitly defines a set-valued mapping $x : [1, 3] \to \mathbb{P}(\boldsymbol{x}) : p \mapsto \{\sqrt[4]{p^2 - 1}, -\sqrt[4]{p^2 - 1}\}$ so that $h(\xi, p) = 0$ for all $p \in [1, 3]$ and $\xi \in x(p)$. While Fig. 6a shows the result using the original bounds, Fig. 6b depicts the effect of using bounds obtained from reverse interval propagation. In the latter case, the reverse interval propagation reduces both $\boldsymbol{x}$ and $\boldsymbol{p}$ to obtain $\tilde{\boldsymbol{x}}$ and $\tilde{\boldsymbol{p}}$. Then, the reverse McCormick propagation is performed using the reduced intervals $\tilde{\boldsymbol{x}}$ and $\tilde{\boldsymbol{p}}$.

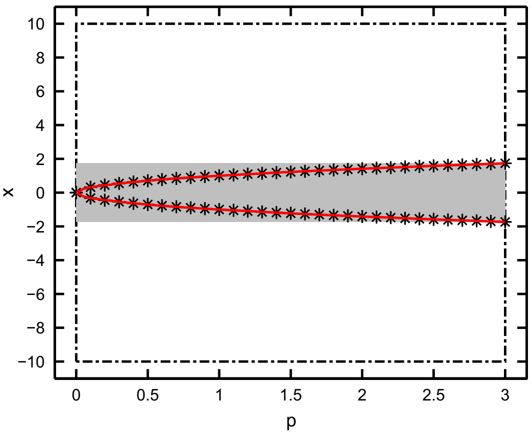

Fig. 5 Result of reverse McCormick propagation for Example 3 showing the original bounds (*dashed-dotted lines*), the improved bounds (*gray box*), the convex and concave relaxations (*solid lines*) as well as results of the set-valued mapping $x(p)$ (*asterisks*)

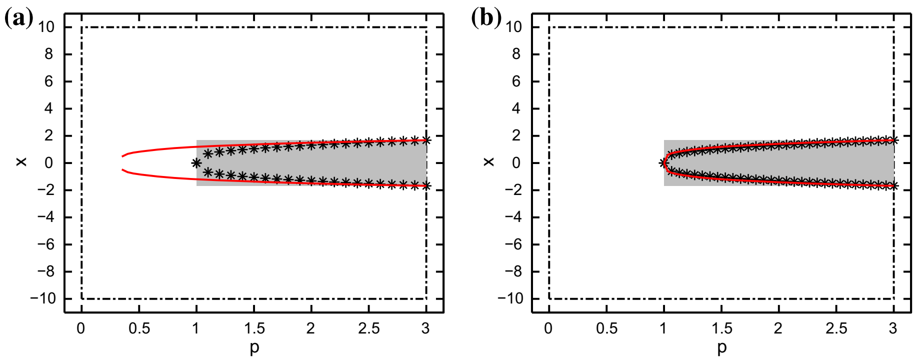

Fig. 6 Result of reverse McCormick propagation for Example 4 showing the original bounds (*dashed-dotted lines*), the improved bounds (*gray box*), the convex and concave relaxations (*solid lines*) as well as results of the set-valued mapping $x(p)$ (*asterisks*). While in **a** the original bounds are used, in **b** the result of the bounds obtained from reverse interval propagation is shown

Example 5 Let $\boldsymbol{x} = [-10, 10]$ and $\boldsymbol{p} = [-3, 3]$. Consider $h(z, p) = z^2 - (\sqrt{p^2 - p} - 2)^4$ with $(z, p) \in \boldsymbol{x} \times \boldsymbol{p}$. Note that $h(z, p) = 0$ implicitly defines a set-valued mapping $x : [-3, 0] \cap [1, 3] \to \mathbb{P}(\boldsymbol{x}) : p \mapsto \{(\sqrt{p^2 - p} - 2)^2, -(\sqrt{p^2 - p} - 2)^2\}$ so that $h(\xi, p) = 0$ for all $p \in [-3, 0] \cap [1, 3]$ and $\xi \in x(p)$. On the other hand, if $p \in (0, 1)$ no feasible z exists that satisfies $h(z, p) = 0$. The results of the reverse McCormick propagation are shown in Fig. 7. Here, the algorithm was supplied with the information that the argument of the square root cannot be negative.

8.2 Inequality constraints

Example 6 Let $\boldsymbol{x} = [-10, 10]$ and $\boldsymbol{p} = [0, 3]$. Consider $h(z, p) = z^2 - p$ and $g(z, p) = (p-1)^2 - z - 2.5$ with $(z, p) \in \boldsymbol{x} \times \boldsymbol{p}$. Note that $h(z, p) = 0$ and $g(z, p) \leq 0$ implicitly defines set-valued mappings $x : [0, 2.03593] \to \mathbb{P}(\boldsymbol{x}) : p \mapsto \{\sqrt{p}, -\sqrt{p}\}$ and $x : (2.03593, 3] \to \mathbb{P}(\boldsymbol{x}) : p \mapsto \{\sqrt{p}\}$ so that $h(\xi, p) = 0$ for all $p \in \boldsymbol{p}$ and $\xi \in x(p)$. However, we are only interested in those (z, p) for which $g(z, p) \leq 0$. The results of the reverse McCormick propagation are shown in Fig. 8.

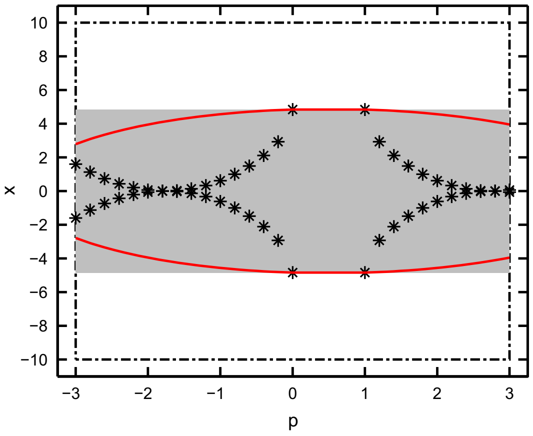

Fig. 7 Result of reverse McCormick propagation for Example 5 showing the original bounds (*dotted lines*), the improved bounds (*dashed lines*), the convex and concave relaxations (*solid lines*) as well as results of the set-valued mapping $x(p)$ (*asterisks*)

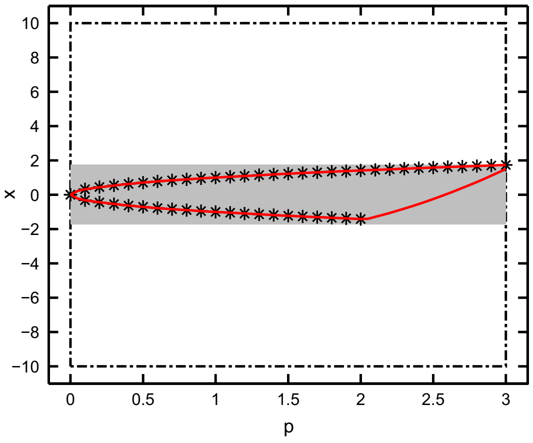

Fig. 8 Result of reverse McCormick propagation for Example 6 showing the original bounds (*dotted lines*), the improved bounds (*gray box*), the convex and concave relaxations (*solid lines*) as well as results of the set-valued mapping $x(p)$ (*asterisks*)

8.3 Objective function

Example 7 Let $\boldsymbol{y} = [-3, 3] \times [-2, 2]^2$ and consider the optimization of the six-hump camel back function [14]

$$\begin{aligned} \min_{y \in \boldsymbol{y}} \quad & f(y) = \left(4 - 2.1y_1^2 + \frac{1}{3}y_1^4\right) y_1^2 + y_1 y_2 + \left(-4 + 4y_2^2\right) y_2^2 \\ \text{s.t.} \quad & g(y) = y_1^2 + (y_2 - 0.5)^2 - 0.5 \le 0 \end{aligned}$$

where an inequality constraint has been added. We are interested in constructing relaxations of $f(y)$ which take the information from the constraint $g(y) \le 0$ into account. Here, let y_1 take the role of the independent and y_2 the role of the dependent variable. The reverse McCormick update will proceed as outlined in Sect. 6.1. Then, one last forward evaluation will be performed to obtain improved relaxations of f. Fig. 9 shows the obtained relaxations. Clearly, the McCormick relaxations can be improved substantially by incorporating the information from the constraint.

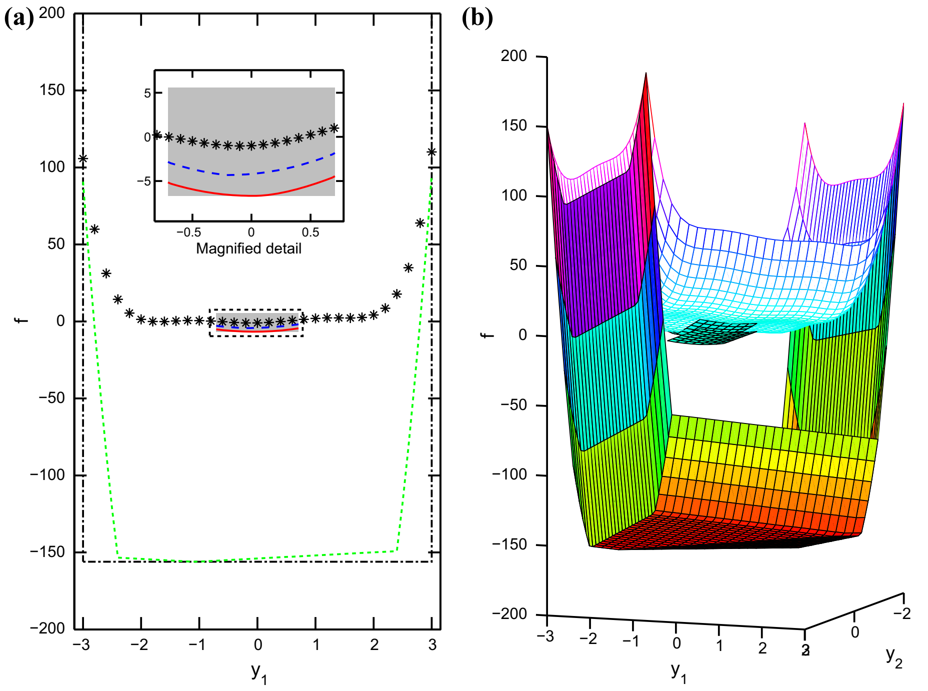

Fig. 9 Result of reverse McCormick propagation for Example 7. In **a** the original bounds (*dashed-dotted line*), the improved bounds (*gray box*), the objective function f (*asterisks*) and the convex relaxations (*solid line*) are shown as well as standard convex McCormick relaxations constructed on $\boldsymbol{y}$ (*short dashed line*) and $\tilde{\boldsymbol{y}}$ (*dashed line*) in a section. In **b** f is shown as a mesh and relaxations are shown as surfaces

9 Global optimization test problems

A set of standard global optimization problems from the COCONUT Benchmark [53] were solved with the reverse McCormick propagation technique to demonstrate its effectiveness. Twenty representative problems involving twenty or less variables, for which the number of variables exceeded the number of equality constraints, were solved from Library 1 of this collection. A basic branch-and-bound framework was used to solve the problems to global optimality, with three different strategies for obtaining lower bounding values for each test case.

As a baseline method, lower bounds on the optimal objective value were found by constructing standard McCormick relaxations of the objective function and constraints on each node, and then solving the resulting nonsmooth convex program. For comparison, reverse McCormick propagation was applied in two different ways to the lower bounding problem: the full-space formulation R2 and the reduced-space formulation R1. In all three cases, a reverse interval propagation step was performed on each node before the relaxations were constructed. This was done both to improve the strength of the lower bound, and in order to most directly show the advantage of applying reverse McCormick propagation beyond that afforded by use of reverse interval propagation. The set of parameters p for each problem was chosen such that the number of remaining dependent variables was equal to the number of equality constraints, and that the resulting square equation system defined by these

Table 1 Set of variables chosen as parameters and the nonsmooth solver used for each of the test problems

Problem	Parametrized variables	Nonsmooth solver
ex4_1_8	x_2	MPBNGC
ex6_1_2	x_2	MPBNGC
mhw4d	x_2, x_3	SolvOpt
ex6_1_4	x_2, x_3	SolvOpt
ex7_2_2	x_1, x_2	SolvOpt
ex8_5_1	x_2, x_3	SolvOpt
ex8_5_2	x_2, x_3	SolvOpt
ex5_2_4	x_1, x_2, x_3, x_4, x_5	SolvOpt
ex9_2_5	x_1	MPBNGC
himmel11	$x_1, x_2, x_3, x_4, x_5, x_6$	MPBNGC
ex9_2_1	x_1	MPBNGC
process	x_1, x_2, x_3	SolvOpt
ex6_1_3	x_2, x_5, x_7	SolvOpt
ex5_2_2	x_3, x_4, x_5, x_6, x_7	SolvOpt
ex7_3_4	$x_9, x_{10}, x_{11}, x_{12}$	MPBNGC
ex9_1_1	x_1	MPBNGC
alkyl	$x_2, x_9, x_{11}, x_{12}, x_{13}, x_{14}, x_{15}$	SolvOpt
ex8_4_5	$x_{12}, x_{13}, x_{14}, x_{15}$	MPBNGC
ex5_4_3	x_{14}, x_{15}, x_{16}	MPBNGC
ex8_4_4	$x_{13}, x_{14}, x_{15}, x_{16}, x_{17}$	SolvOpt

The variable names correspond to those given in the AMPL model file from the COCONUT library

constraints was structurally nonsingular. Since such a partitioning is non-unique in general, different parameter sets may exist for each test problem that could perform either more or less favorably than those explored in this work. Table 1 shows the variables chosen as parameters for each of the test problems.

In practice, solving the convex lower bounding problems was found to be highly nontrivial due to the nonsmooth nature of the McCormick relaxations and the presence of constraints in two of the formulations. Two freely-available nonsmooth local optimization algorithms were used in this work: MPBNGC, an implementation of the proximal bundle method [36], and SolvOpt, an implementation of Shor's r-algorithm with an exact penalty formulation for handling constraints [29]. It was observed that neither of these algorithms could be used on all problems in the test set, either due to the assumptions of the method or numerical difficulties. However, for each problem, at least one of the two algorithms was suitable, and the same solver was used for all three formulations for consistency. The solver used for each problem is also noted in Table 1. It was noted that MPBNGC generally required fewer but more expensive iterations than SolvOpt to solve the lower bounding problems. Improved methods for constrained nonsmooth optimization using reverse McCormick propagation will be a topic of future research.

The SQP algorithm SNOPT [18] was used to obtain upper bounds and find feasible points required for initialization of the MPBNGC method. No range reduction techniques were employed in this branch-and-bound implementation, and the relative and absolute tolerances used to terminate the algorithm were 10^{-3} and 10^{-8}, respectively. Branching was performed such that the current box was bisected along the largest current width relative to the original

Table 2 Results of applying reverse propagation of McCormick relaxations to a set of standard global optimization test problems

	Standard relaxations		Formulation (R2)		Formulation (R1)	
Problem	Time (s)	Iterations	Time (s)	Iterations	Time (s)	Iterations
ex4_1_8	0.01425	1	0.02603	1	0.004586	1
ex6_1_2	183.8	613	0.9245	115	0.1499	22
mhw4d	2.804	61	0.3540	23	0.06245	11
ex6_1_4	7386	15,343	8.986	279	1.034	111
ex7_2_2	234.7	277	5.572	77	0.8964	398
ex8_5_1	62.80	2,809	143.4	4,219	40.31	3,443
ex8_5_2	591.1	9,774	33.56	909	11.13	3,993
ex5_2_4	0.1968	95	0.2111	83	0.09289	44
ex9_2_5	61.50	109	2.820	45	1.469	71
himmel11	186.9	251	13.95	78	2.357	123
ex9_2_1	83.96	103	18.25	65	3.383	104
process	285.6	107	212.4	583	841.2	28,745
ex6_1_3	2006	14,227	133.9	2,815	187.0	7,555
ex5_2_2	6462	36,621	805.7	15,085	26.45	3,490
ex7_3_4	1.738×10^5	836,683	8.113×10^4	173,014	1219	52,486
ex9_1_1	111.8	63	0.2886	1	0.01650	1
alkyl	151.0	37	1851	3,331	42.90	2,158
ex8_4_5	1.238×10^4	63,595	2,701	10,181	7.425	2,423
ex5_4_3	903.8	2,135	20.39	323	2.110	571
ex8_4_4	1.493×10^5	399,809	6.518×10^4	134,005	2.974×10^4	673,281

Problems are listed in order of increasing number of variables

box dimensions. Nodes were selected according to the lowest lower bound heuristic. The results of the numerical tests can be found in Table 2.

The results indicate the use of reverse McCormick propagation significantly improves solution speed (in both elapsed time and iteration count) on the majority of the test problems. The full-space (R2) formulation seems to be most effective in reducing the number of branch-and-bound iterations required for convergence, however, since it requires the solution of a difficult nonsmooth local optimization problem at each iteration, the cost per iteration is high. The reduced-space (R1) formulation was generally the fastest to converge in terms of elapsed time, although sometimes required more branch-and-bound iterations than either the full-space reverse mode or standard formulations. This is likely due to the fact that the two full-space formulations are guaranteed to be quadratically convergent methods, whereas the reduced-space formulation is not [10]. However, the reduced number of decision variables in the problem offsets the effect of the lower convergence order in most cases. Furthermore, we note that there is no reason to even expect that the reduced-space formulation will converge for an arbitrary partition (x, p). It appears necessary that for all $p \in \boldsymbol{p}$ either $X(p)$ is a singleton or $X(p) = \emptyset$, but a derivation of a convergence result for reverse interval propagation would also be required. These conditions were not checked *a priori* for these test cases, and Table 1 shows the results from those problems which did successfully converge in the (R1) formulation. We note that other problems from the library were tested for which the (R2) formulation converged but the (R1) formulation did not, although numerical difficulties with the nonsmooth solvers were also observed in some of these cases.

In a few cases, the reverse McCormick propagation was observed to be less effective than using the Standard McCormick relaxations (e.g. problems process and alkyl). These exceptions occur when the reverse McCormick update fails and returns the empty set during the lower bounding procedure on a large number of iterations, which can happen when the local optimization routine picks a value of p for which no $X(p)$ exists and no relaxations can be calculated (see Fig. 6a when $p \leq \frac{1}{3}$ for a graphical example). In the current implementation, this is handled by using the information from the reverse interval pass to calculate the lower bounding value, and then immediately branching on the node. Handling this case more effectively is the topic of ongoing research, and will further improve the performance of this promising constraint propagation method.

10 Conclusion

Reverse McCormick propagation, a new method to construct and improve McCormick relaxations of implicitly defined set-valued mappings has been presented. It takes advantage of the directed acyclic graph representation of a factorable function, which has been previously used for interval calculations [50,62]. Bounds and relaxations of factors can often be improved by using information about the permissible range of a factorable function and propagating it backwards through the graph. In particular, this allows the construction and improvement of relaxations of mappings that are only implicitly defined. This is useful in the context of CSPs since it allows to construct convex relaxations of non-convex solution sets defined by nonlinear equality constraints and non-convex inequality constraints. Furthermore, McCormick relaxations of the objective function of an NLP can be improved using information contained in the constraints. While Stuber et al. [55] also put forward methods to construct relaxations of implicit functions, the method presented here does not require existence nor uniqueness of the implicit function on all or parts of the domain. Furthermore, it is less computationally intensive and does not require a pre-processing step. It also provides a reduced-space relaxation for nonconvex programs that can take constraints into account, but does not require convex optimizers that can cope with general nonsmooth nonlinear constraints. When used as a constraint propagation method in the context of global optimization, the reverse McCormick approach proved to be effective at improving solution rate compared to the standard McCormick relaxation approach and reverse interval propagation over a set of representative test problems.

References

1. Adjiman, C.S., Floudas, C.A.: Rigorous convex underestimators for general twice-differentiable problems. J. Glob. Optim. **9**, 23–40 (1996)
2. Barták, R.: Theory and practice of constraint propagation. In: Proceedings of the 3rd Workshop on Constraint Programming in Decision and Control, pp. 7–14 (2001)
3. Belotti, P., Lee, J., Liberti, L., Margot, F., Wächter, A.: Branching and bounds tightening techniques for non-convex MINLP. Optim. Methods Softw. **24**(4–5), 597–634 (2009)
4. Belotti, P., Cafieri, S., Lee, J., Liberti, L.: Feasibility-based bounds tightening via fixed points. In: Wu, W., Daescu, O. (eds.) Combinatorial Optimization and Applications, vol. 6508, pp. 65–76. Springer, Berlin (2010)

5. Benhamou, F., Older, W.J.: Applying interval arithmetic to real, integer, and boolean constraints. J. Logic Program. **32**(1), 1–24 (1997)
6. Benhamou, F., McAllester, D., Van Hentenryck, P.: CLP (intervals) revisited. In: Bruynooghe, M. (ed.) Proceedings of the 1994 International Symposium on Logic Programming, pp. 124–138. MIT Press, Cambridge (1994)
7. Benhamou, F., Goualard, F., Granvilliers, L., Puget, J.F.: Revising hull and box consistency. In: Proceedings of the International Conference on Logic Programming (ICLP'99), pp. 230–244. MIT Press, Cambridge (1999)
8. Benhamou, F., Granvilliers, L., Goualard, F.: Interval constraints: results and perspectives. In: New Trends in Constraints, Lecture Notes in Artificial Intelligence vol. 1865, pp. 1–16. Springer, Berlin (2000)
9. Bessiere, C.: Constraint propagation. In: Rossi, F., van Beek, P., Walsh, T. (eds.) Handbook of Constraint Programming, pp. 29–83. Elsevier, Amsterdam (2006). chap 3
10. Bompadre, A., Mitsos, A.: Convergence rate of McCormick relaxations. J. Glob. Optim. **52**, 1–28 (2012)
11. Chachuat, B.: MC++—A Versatile Library for McCormick Relaxations and Taylor Models. http://www3.imperial.ac.uk/people/b.chachuat/research/ (2011)
12. Cleary, J.G.: Logical arithmetic. Future Comput. Syst. **2**(2), 124–149 (1987)
13. Davis, E.: Constraint propagation with interval labels. Artif. Intell. **32**(3), 281–331 (1987)
14. Dixon, L.C.W., Szego, G.P.: The optimization problem: an introduction. In: Dixon, L.C.W., Szego, G.P. (eds.) Towards Global Optimization. North Holland, New York (1978)
15. Domes, F., Neumaier, A.: Constraint propagation on quadratic constraints. Constraints **15**(3), 404–429 (2010)
16. Dulmage, A., Mendelsohn, N.: Coverings of bipartite graphs. Can. J. Math. 517–534 (1958)
17. Falk, J.E., Soland, R.M.: An algorithm for separable nonconvex programming problems. Manag. Sci. **15**, 550–569 (1969)
18. Gill, P.E., Murray, W., Saunders, M.A.: SNOPT: an SQP algorithm for large-scale constrained optimization. SIAM Rev. **47**(1), 99–131 (2002)
19. Granvilliers, L., Benhamou, F.: Algorithm 852: RealPaver: an interval solver using constraint satisfaction techniques. ACM Trans. Math. Softw. **32**(1), 138–156 (2006)
20. Griewank, A., Walther, A.: Evaluating Derivatives, 2nd edn. Society for Industrial and Applied Mathematics, Philadelphia (2008)
21. Hansen, E., Walster, G.W.: Global Optimization Using Interval Analysis, 2nd edn. Marcel Dekker, New York (2004)
22. Hansen, P., Jaumard, B., Lu, S.H.: An analytical approach to global optimization. Math. Program. **52**(1–3), 227–254 (1991)
23. Hiriart-Urruty, J.B., Lemaréchal, C.: Convex Analysis and Minimization Algorithms. Springer, Berlin (1993)
24. Hooker, J.: Logic-Based Methods for Optimization: Combining Optimization and Constraint Satisfaction. Wiley, New York (2000)
25. Horst, R., Tuy, H.: Global Optimization: Deterministic Approaches, 3rd edn. Springer, Berlin (1996)
26. Hyvönen, E.: Constraint reasoning based on interval arithmetic: the tolerance propagation approach. Artif. Intell. **58**(1–3), 71–112 (1992)
27. Jaulin, L.: Solving set-valued constraint satisfaction problems. Computing **94**(2–4), 297–311 (2012)
28. Jaulin, L., Michel, K., Didrit, O., Walter, E.: Applied Interval Analysis. Springer, London (2001)
29. Kappel, F., Kuntsevich, A.: An implementation of Shor's r-algorithm. Comput. Optim. Appl. **15**, 193–205 (2000)
30. Kearfott, R.B., Nakao, M., Neumaier, A., Rump, S.M., Shary, S., Van Hentenryck, P.: Standardized notation in interval analysis. In: Proceedings of the XIII Baikal International School-seminar "Optimization methods and their applications", vol. 4, pp. 106–113. Institute of Energy Systems SB RAS, Irkutsk, Russia (2005)
31. Kiwiel, K.C.: Methods of Descent for Nondifferentiable Optimization. Springer, Berlin (1985)
32. Lebbah, Y., Michel, C., Rueher, M., Daney, D., Merlet, J.P.: Efficient and safe global constraints for handling numerical constraint systems. SIAM J. Numer. Anal. **42**(5), 2076 (2005)
33. Lhomme, O.: Consistency techniques for numeric CSPs. In: International Joint Conference on Artificial Intelligence, pp. 232–238 (1993)
34. Lukšan, L., Vlček, J.: Algorithm 811: NDA: algorithms for nondifferentiable optimization. ACM Trans. Math. Softw. **27**, 193–213 (2001)
35. Mackworth, A.K.: Consistency in networks of relations. Artif. Intell. **8**(1), 99–118 (1977)
36. Mäkelä, M.M.: Multiobjective Proximal Bundle Method for Nonconvex Nonsmooth Optimization: Fortran Subroutine MPBNGC 2.0. Reports of the Department of Mathematical Information Technology, Series B, Scientific computing (B 13/2003), University of Jyväskylä, Jyväskylä (2003)

37. Mäkelä, M.M., Neittaanmäki, P.: Nonsmooth Optimization. World Scientific, Singapore (1992)
38. MATLAB.: Version 8.3.0 (R2014a). The MathWorks Inc., Natick, MA (2014)
39. McCormick, G.P.: Computability of global solutions to factorable nonconvex programs: part I—convex underestimating problems. Math. Program. **10**, 147–175 (1976)
40. Melquiond, G., Pion, S., Brönnimann, H.: Boost Interval Arithmetic Library. http://www.boost.org/doc/libs/1_49_0/ (2006)
41. Mitsos, A., Chachuat, B., Barton, P.I.: McCormick-based relaxations of algorithms. SIAM J. Optim. **20**, 573–601 (2009)
42. Moore, R.E.: Methods and Applications of Interval Analysis. SIAM, Philadelphia (1979)
43. Neumaier, A.: Interval Methods for Systems of Equations. Cambridge University Press, Cambridge (1990)
44. Neumaier, A.: Complete search in continuous global optimization and constraint satisfaction. In: Iserles, A. (ed.) Acta Numerica, vol. 13, pp. 271–370. Cambridge University Press, Cambridge (2004)
45. Ratschek, H., Rokne, J.: Computer Methods for the Range of Functions. Ellis Horwood, Chichester (1984)
46. Rudin, W.: Principles of Mathematical Analysis, 3rd edn. McGraw-Hill, New York (1976)
47. Ryoo, H.S., Sahinidis, N.V.: Global optimization of nonconvex NLPs and MINLPs with applications in process design. Comput. Chem. Eng. **19**(5), 551–566 (1995)
48. Sahinidis, N.V., Tawarmalani, M.: BARON Solver Manual. http://gams.com/dd/docs/solvers/baron.pdf (2009)
49. Sam-Haroud, D., Faltings, B.: Consistency techniques for continuous constraints. Constraints **1**(1–2), 85–118 (1996)
50. Schichl, H., Neumaier, A.: Interval analysis on directed acyclic graphs for global optimization. J. Glob. Optim. **33**(4), 541–562 (2005)
51. Scott, J.K.: Reachability Analysis and Deterministic Global Optimization of Differential-algebraic Systems. Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, MA (2012)
52. Scott, J.K., Stuber, M.D., Barton, P.I.: Generalized McCormick relaxations. J. Glob. Optim. **51**(4), 569–606 (2011)
53. Shcherbina, O., Neumaier, A., Sam-Haroud, D., Vu, X.H., Nguyen, T.V.: Benchmarking global optimization and constraint satisfaction codes. In: Bliek, C., Jermann, C., Neumaier, A. (eds.) Global Optimization and Constraint Satisfaction, pp. 211–222. Springer, Berlin (2003)
54. Smith, E.M.B., Pantelides, C.C.: Global optimisation of nonconvex MINLPs. Comput. Chem. Eng. **21**, S791–S796 (1997)
55. Stuber, M.D., Scott, J.K.: Convex and concave relaxations of implicit functions. Optim. Methods Softw. (in press) (2014)
56. Tawarmalani, M., Sahinidis, N.V.: Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming. Kluwer, Dordrecht (2002)
57. Tawarmalani, M., Sahinidis, N.V.: Global optimization of mixed-integer nonlinear programs: a theoretical and computational study. Math. Program. **99**, 563–591 (2004)
58. Tawarmalani, M., Sahinidis, N.V.: A polyhedral branch-and-cut approach to global optimization. Math. Program. **103**, 225–249 (2005)
59. Van Hentenryck, P., McAllester, D., Kapur, D.: Solving polynomial systems using a branch and prune approach. SIAM J. Numer. Anal. **34**(2), 797–827 (1997)
60. Van Hentenryck, P., Michel, L., Benhamou, F.: Constraint programming over nonlinear constraints. Sci. Comput. Program. **30**(1–2), 83–118 (1998)
61. Vu, X.H., Sam-Haroud, D., Silaghi, M.C.: Numerical constraint satisfaction problems with non-isolated solutions. In: Bliek, C., Jermann, C., Neumaier, A. (eds.) Global Optimization and Constraint Satisfaction, Lecture Notes in Computer Science, vol. 2861, pp. 194–210. Springer, Berlin (2003)
62. Vu, X.H., Schichl, H., Sam-Haroud, D.: Interval propagation and search on directed acyclic graphs for numerical constraint solving. J. Glob. Optim. **45**(4), 499–531 (2009)