Journal of Symbolic Computation 47 (2012) 1416–1446

Contents lists available at SciVerse ScienceDirect

Journal of Symbolic Computation

journal homepage: www.elsevier.com/locate/jsc

Abstract interpretation meets convex optimization☆,☆☆

Thomas Martin Gawlitza[a,b,1], Helmut Seidl[c], Assalé Adjé[d], Stéphane Gaubert[e], Éric Goubault[f]

[a] *VERIMAG, Grenoble, France*
[b] *School of Information Technologies, The University of Sydney, Sydney, Australia*
[c] *Technische Universität München, München, Germany*
[d] *CEA, LIST and LIX, Ecole Polytechnique (MeASI), France*
[e] *INRIA Saclay and CMAP, Ecole Polytechnique, F-91128 Palaiseau Cedex, France*
[f] *CEA, LIST (MeASI), F-91191 Gif-sur-Yvette Cedex, France*

ARTICLE INFO

Article history:
Received 11 June 2011
Accepted 13 September 2011
Available online 24 December 2011

Keywords:
Static analysis
Abstract interpretation
Fixpoint equations
Convex optimization
Strategy improvement algorithms
Semi-definite relaxation

ABSTRACT

Numerical static program analyses by abstract interpretation, e.g., the problem of inferring bounds for the values of numerical program variables, are faced with the problem that the abstract domains often contain infinite ascending chains. In order to enforce termination within the abstract interpretation framework, a widening/narrowing approach can be applied that trades the guarantee of termination against a potential loss of precision. Alternatively, recently *strategy improvement algorithms* have been proposed for computing numerical invariants which do not suffer the imprecision incurred by widenings. Before, strategy improvement algorithms have successfully been applied for solving two-players zero-sum games. In this article we discuss and compare max-strategy and min-strategy improvement algorithms for static program analysis. For that, the algorithms are cast within a common general framework of solving systems of fixpoint equations $\mathbf{x} = e$ where the right-hand sides e are maxima of finitely many *monotone* and *concave* functions. Then we indicate how the general setting can be instantiated for inferring numerical invariants of programs based on non-linear templates.

☆ This work was partially funded by the ANR project ASOPT.

☆☆ VERIMAG is a joint laboratory of CNRS, Université Joseph Fourier and Grenoble INP.

E-mail addresses: gawlitza@it.usyd.edu.au, Thomas.Gawlitza@imag.fr (T.M. Gawlitza), seidl@in.tum.de (H. Seidl), Assale.Adje@cea.fr (A. Adjé), Stephane.Gaubert@inria.fr (S. Gaubert), Eric.Goubault@cea.fr (É. Goubault).

[1] Tel.: +33 4 56 52 03 86; fax: +33 4 56 52 03 44.

0747-7171/$ – see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.jsc.2011.12.048

1. Introduction

Mathematical optimization aims at finding a value within an area of *feasible* values which maximizes (resp. minimizes) a given objective function. Quite efficient techniques have been developed for particular cases that are important in practice, e.g., when the objective function is linear and the area of feasible values a convex polytope (*linear programming*, see e.g. Schrijver, 1986) or even an intersection of a convex polytope with the positive semi-definite cone (*semi-definite programming*, see e.g. Todd, 2001) or a convex set that is defined through convex constraints (*convex optimization*, see e.g. Boyd and Vandenberghe, 2004; Nemirovski, 2005). In a certain sense, also numerical static program analysis based on abstract interpretation can often be cast as an optimization problem as follows: Assume that we are given a complete lattice of potential program invariants at program points, i.e., an abstract domain. Then, each control-flow edge from a program point u to a program point v induces constraints on the invariants for u and v. These constraints describe the feasible area. The objective of the analysis is to minimize all invariants for the program points.

In general, it is not clear how this insight may lead to better algorithms. In this article, however, we show that in the case of *template-based* analysis of relational numerical properties, techniques from mathematical optimization allow to construct novel program analysis algorithms. The *templates* we consider are (multivariate) polynomials in the program variables such as $2x_1^2 + 3x_2^2 + 2x_1x_2$, where x_1 an x_2 are program variables. The goal of the analysis is to determine, for every program point v, a safe upper bound to each template when reaching that program point v. In order to be as precise as possible, this upper bound should be as small as possible. Different templates may serve different purposes. If the analysis is only meant to infer (decently small) *intervals* for the values of the program variables $x_1, \ldots, x_n$, templates of the form x_i and $-x_i$ may suffice. If the analysis additionally should infer bounds on the *differences* between certain variables, templates of the form $x_i - x_j$ should be used.

Templates consisting of arbitrary *linear* combinations have been introduced and studied by Sankaranarayanan et al. (2005). In some cases, e.g., when trying to prove that certain linear filters do not lead to floating-point overflows, linear templates are not sufficient (see e.g. Feron and Alegre, 2008b). However, these cases can be treated by using *quadratic* templates (see e.g. Feron and Alegre, 2008a; Adjé et al., 2010).

In this article, instead of directly performing the template based analysis, we reduce the static program analysis problem to the problem of computing the least solution of

$$\mathbf{x} \geq f(\mathbf{x}), \tag{1}$$

where the unknown $\mathbf{x}$ now may take values in $\overline{\mathbb{R}}^n$, where $\overline{\mathbb{R}} := \mathbb{R} \cup \{-\infty, \infty\}$. In other words: we are now interested in computing the least fixpoint μf of f. The components of the variable $\mathbf{x}$ are the upper bounds to the templates at the different program points. In our application, the right-hand side f turns out to be a point-wise maximum of finitely many *monotone and concave* operators on $\overline{\mathbb{R}}^n$.

Strategy iteration techniques can be applied to compute invariants for template domains where all templates are linear combinations of program variables. In the simple case of intervals, these approaches allow to perform interval analysis without widening (Costan et al., 2005; Gawlitza and Seidl, 2007a). In case of more complex linear combinations, arbitrary *template polyhedra domains* (Sankaranarayanan et al., 2005) such as the *octagon domain* (Miné, 2001) can be handled (Gaubert et al., 2007; Gawlitza and Seidl, 2007b). For quadratic templates, the above strategy iteration approaches can be utilized for computing (resp. approximating) a *semi-definite relaxation* of the abstract semantics (cf. Gawlitza and Seidl, 2010; Adjé et al., 2010).

We present two *strategy improvement approaches* for computing respectively approximating the least solution to (1).

The Min-Strategy Iteration Approach. The min-strategy iteration approach as advocated by Adjé et al. (2010) works as follows: conceptually, the first step is to choose a (potentially infinite) set Π of min-strategies. A min-strategy $\pi \in \Pi$ is a monotone operator on $\overline{\mathbb{R}}^n$ that over-approximates f, i.e., $\pi(x) \geq f(x)$ for all $x \in \overline{\mathbb{R}}^n$. We moreover require that, for each $x \in \overline{\mathbb{R}}^n$, we are able to select a min-strategy $\pi \in \Pi$ such that $f(x) = \pi(x)$. In other words, we require

that

$$f(x) = \min\{\pi(x) \mid \pi \in \Pi\} \quad \text{for all } x \in \overline{\mathbb{R}}^n. \tag{2}$$

To decompose the operator f in such a way makes sense, if the problem of computing the least fixpoint $\mu\pi$ of a min-strategy $\pi \in \Pi$ is simpler than computing the least fixpoint μf of f. The method then makes use of the fact that

$$\mu f = \min\{\mu\pi \mid \pi \in \Pi\}. \tag{3}$$

For the case we consider in this article, we choose Π to contain all operators that over-approximate f and are point-wise maxima of finitely many monotone and affine operators. Least fixpoints of such operators can be efficiently computed using linear programming (cf. Gaubert et al., 2007).

For the particular case we are studying in this article, the min-strategy iteration approach works similar to Newton's method. It starts with some solution $x^{(0)}$ of (1) and constructs a *decreasing sequence* $(x^{(k)})_{k\in\mathbb{N}}$ of solutions. For any solution $x^{(k)}$, the next solution $x^{(k+1)}$ is obtained as follows: we select a min-strategy π such that $\pi(x^{(k)}) = f(x^{(k)})$ holds (such a min-strategy must exist). The min-strategy is the pendant to the first order Taylor approximation used within Newton's method. As we will see, this guarantees that the sequence $(x^{(k)})_{k\in\mathbb{N}}$ is decreasing. The solution $x^{(k+1)}$ is then obtained as the least fixpoint $\mu\pi$ of π which can be computed efficiently by means of linear programming (cf. Gaubert et al., 2007). The crucial step here is to determine a min-strategy π with $\pi(x^{(k)}) = f(x^{(k)})$. As we will see, this problem can be tackled using convex optimization.

The min-strategy improvement algorithm improves a solution step by step. Each solution $x^{(k)}$ is a safe over-approximation to the least solution and the sequence $(x^{(k)})_{k\in\mathbb{N}}$ is decreasing. However, in the general case, the method is neither guaranteed to terminate, nor is the sequence $(x^{(k)})_{k\in\mathbb{N}}$ guaranteed to converge to the least solution. It can be get stuck in a local minimum. One of the most important advantages of the min-strategy improvement algorithm is that it can be stopped at any time with a safe over-approximation to the least solution. Another advantage is that the convex optimization problems that have to be solved are quite small compared to the ones we have to solve when we apply max-strategy iteration (see below).

The Max-Strategy Iteration Approach. The max-strategy improvement algorithm of Gawlitza and Seidl (2010) computes the least solution of (1) by iterating over max-strategies. Recall that f is a point-wise maximum of finitely many monotone and concave operators. Hence, we can find a *finite* set Σ of monotone and concave operators such that

$$f(x) = \max\{\sigma(x) \mid \sigma \in \Sigma\} \quad \text{for all } x \in \overline{\mathbb{R}}^n. \tag{4}$$

Each function $\sigma \in \Sigma$ is a max-strategy. Observe that Equality (4) in particular implies that, for each x, there exists some max-strategy $\sigma \in \Sigma$ such that $f(x) = \sigma(x)$. In the cases we consider in this article, Σ is finite. It contains at most exponentially many max-strategies (exponential in the size of the representation of f).

The algorithm constructs a sequence $(\sigma^{(k)})_{k\in\mathbb{N}}$ of max-strategies and a *strictly increasing sequence* $(x^{(k)})_{k\in\mathbb{N}}$ of approximates to the least solution. It starts with a max-strategy $\sigma^{(0)}$ and an approximate $x^{(0)} \in \overline{\mathbb{R}}^n$ with $x^{(0)} \leq \sigma^{(0)}(x^{(0)}) \leq \mu f$. Now assume that, after performing k improvement steps, we have a max-strategy $\sigma^{(k)}$ and an approximate $x^{(k)}$ at hand such that $x^{(k)} \leq \sigma^{(k)}(x^{(k)}) \leq \mu f$. In order to determine $x^{(k+1)}$, we conceptually perform a least fixpoint iteration using the current max-strategy $\sigma^{(k)}$, where we start at the approximate $x^{(k)}$. That is, $x^{(k+1)}$ is the least fixpoint of $\sigma^{(k)}$ that is greater than or equal to $x^{(k)}$. For the cases we are studying within this article, $x^{(k+1)}$ can be computed using convex optimization techniques, provided that we follow some rules. The next max-strategy $\sigma^{(k+1)}$ can be chosen as some max-strategy such that $\sigma^{(k+1)}(x^{(k+1)}) = f(x^{(k+1)})$. This guarantees a progress in each iteration.

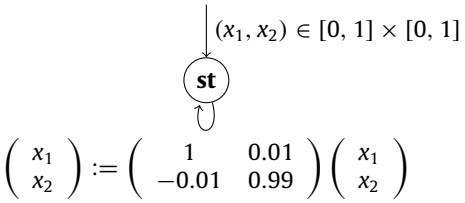


Fig. 1. The harmonic oscillator.

Regarding the applications we study in this article, the advantage of the max-strategy iteration approach (compared to the min-strategy iteration approach) is that it terminates after at most exponentially many steps. Moreover, it returns the precise result. That is, it always returns the least solution of (1). A disadvantage is that only when the least solution is reached, a *safe* solution is obtained.

This article is organized as follows: In Section 2, we discuss an introductory example, where an analysis based on linear templates only infers trivial invariants. Non-trivial invariants, though, can be obtained with quadratic templates and a semi-definite relaxation of the resulting abstract semantics. In Section 3, we discuss how an abstract semantics should be relaxed such that the resulting *relaxed semantic equations* fit in our framework. After introducing basic notations in Section 4, we explain the min-strategy iteration approach in Section 5 and the max-strategy iteration approach in Section 6. In this article, we do not aim at completeness. Instead, we focus on those technical details which are directly connected with our application. Section 7 is dedicated to a comparison of the two approaches and some concluding remarks.

2. Motivation and running example

In this section we take a look at the harmonic oscillator example of Adjé et al. (2010). The program (here given as a C code snippet) consists of the following simple loop:

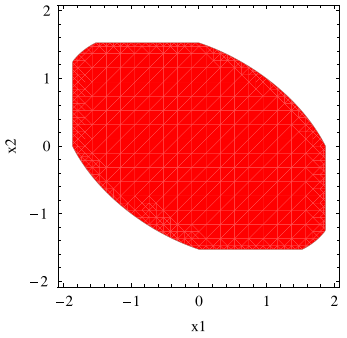
```
float x_1, x_2, tmp;
x_1 = rand();
x_2 = rand();
while ( TRUE ) {
  printf("%f, %f\n", x_1, x_2);
  tmp = 1. * x_1 + 0.01 * x_2;
  x_2 = -0.01 * x_1 + 0.99 * x_2;
  x_1 = tmp;
}
```

Here, we assume that rand() returns a random float value in the interval $[0, 1]$. Fig. 1 shows the control-flow graph of the program. Here, for simplicity, we assume that all program variables are real-valued, i.e., we from now on consider floats as reals. The program implements an Euler explicit scheme with a small step $h = 0.01$, i.e., it simulates the linear system

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \leftarrow \begin{pmatrix} 1 & h \\ -h & 1-h \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}. \tag{5}$$

The invariant found with our strategy improvement methods (see Sections 5 and 6) is shown in Fig. 2. For finding this invariant, we aim to compute upper bounds $b_1, \ldots, b_5 \in \overline{\mathbb{R}} = \mathbb{R} \cup \{-\infty, \infty\}$ that are as small as possible and fulfill the following inequalities for all possible values of the program variables x_1 and x_2 at program point **st**:

$$-x_1 \le b_1 \qquad x_1 \le b_2 \qquad -x_2 \le b_3 \qquad x_2 \le b_4 \qquad 2x_1^2 + 3x_2^2 + 2x_1x_2 \le b_5. \tag{6}$$



$$-1.8708 \leq x_1 \leq 1.8708 \text{ and } -1.5275 \leq x_2 \leq 1.5275 \text{ and } 2x_1^2 + 3x_2^2 + 2x_1x_2 \leq 7$$

Fig. 2. Invariants for the Harmonic Oscillator.

This means that we consider a domain which maintains upper bounds for the linear polynomials $-x_1, x_1, -x_2, x_2$ (i.e., we consider intervals for the values of the program variables) and the *non-linear* polynomial $2x_1^2 + 3x_2^2 + 2x_1x_2$. The last polynomial comes from the Lyapunov function that the designer of the algorithm may have considered to prove the stability of his scheme, *before it has been implemented*. In view of proving the implementation correct, one is naturally led to considering such polynomial templates.[2] Let us finally remark that the loop invariant obtained when using intervals, zones, octagons or even polyhedra (hence with any set of linear templates) is the trivial invariant $\top$ (the value of the program variables x_1 and x_2 cannot be bounded).

The benchmarks of Adjé et al. (2010) and Gawlitza and Seidl (2010) include a computation of invariants of the same quality as for the harmonic oscillator for an implementation of the Arrow–Hurwicz algorithm. This is essentially an harmonic oscillator limited by a non-linear saturation term (a projection on the positive cone). The benchmarks also include a symplectic integration scheme. This is a highly degenerated example for which alternative methods fail due to the absence of stability margins.

3. Abstract interpretation and monotone fixpoint equations

In this section we consider static program analysis by abstract interpretation as introduced by Cousot and Cousot (1977) for the particular case of template based numerical properties, and reduce the inference of corresponding program invariants to solving systems of inequalities of the form $\mathbf{x} \geq e$ over $\overline{\mathbb{R}} = \mathbb{R} \cup \{-\infty, \infty\}$, where the right-hand sides e are *monotonic* and *concave*.

3.1. Notations

For a function $f : X \to Y$ and a subset X' of X, $f|_{X'}$ denotes the restriction of f to X', i.e., the function $f|_{X'} : X' \to Y$ is defined by $f|_{X'}(x') = f(x')$ for all $x' \in X'$.

The set of real numbers is denoted by $\mathbb{R}$. The complete linear ordered set $\mathbb{R} \cup \{-\infty, \infty\}$ is denoted by $\overline{\mathbb{R}}$. For $f : X \to \overline{\mathbb{R}}^m$ with $X \subseteq \overline{\mathbb{R}}^n$, we set

$$\mathrm{dom}(f) := \{x \in X \mid f(x) \in \mathbb{R}^m\}, \quad \text{and} \quad \mathrm{fdom}(f) := \mathrm{dom}(f) \cap \mathbb{R}^n. \tag{7}$$

We denote the i-th row (resp. j-th column) of a matrix A by $A_{i\cdot}$ (resp. $A_{\cdot j}$). Accordingly, $A_{i,j}$ denotes the entry in the i-th row and the j-th column. We also use this notation for vectors and functions $f : X \to Y^k$, i.e., $f_{i\cdot}(x) = (f(x))_{i\cdot}$ for all $x \in X$ and all $i \in \{1, \ldots, k\}$.

[2] Of course, as for the linear templates of Sankaranarayanan et al. (2005, 2006), we are interested in automatically finding or refining the set of polynomial templates considered to achieve good precision. This, however, is outside the scope of this article.

For $x, y \in \overline{\mathbb{R}}^n$, we write $x \leq y$ iff $x_{i\cdot} \leq y_{i\cdot}$ for all $i \in \{1, \ldots, n\}$. $\overline{\mathbb{R}}^n$ is partially ordered by $\leq$. We write $x < y$ iff $x \leq y$ and $x \neq y$. Finally, we write $x \lhd y$ iff $x_{i\cdot} < y_{i\cdot}$ for all $i \in \{1, \ldots, n\}$. The elements x and y are called *comparable* iff $x \leq y$ or $y \leq x$.

Let $\mathbb{D}$ be a partially ordered set. We denote the *least upper bound* and the *greatest lower bound* of a set $X \subseteq \mathbb{D}$ by $\bigvee X$ and $\bigwedge X$, respectively, provided that they exist. The existence is in particular guaranteed if $\mathbb{D}$ is a *complete lattice*. The least element $\bigvee \emptyset$ (resp. the greatest element $\bigwedge \emptyset$) is denoted by $\bot$ (resp. $\top$), provided that it exists. Accordingly, we define the binary operators $\vee$ and $\wedge$ by

$$x \vee y := \bigvee \{x, y\} \quad \text{and} \quad x \wedge y := \bigwedge \{x, y\} \tag{8}$$

for all $x, y \in \mathbb{D}$, respectively. If $\mathbb{D}$ is a *linearly ordered set* (for instance $\mathbb{R}$ or $\overline{\mathbb{R}}$), then $\vee$ is the *maximum* operator and $\wedge$ the *minimum* operator. For $\square \in \{\vee, \wedge\}$, we will also consider $x_1 \square \cdots \square x_k$ as the application of a k-ary operator. This will cause no problems, since the binary operators $\vee$ and $\wedge$ are associative and commutative.

A function $f : \mathbb{D}_1 \to \mathbb{D}_2$, where $\mathbb{D}_1$ and $\mathbb{D}_2$ are partially ordered sets, is called *monotone* iff $x \leq y \implies f(x) \leq f(y)$ for all $x, y \in \mathbb{D}_1$.

3.2. Convex and concave functions

A set $X \subseteq \mathbb{R}^n$ is called *convex* iff $\lambda x + (1-\lambda)y \in X$ for all $x, y \in X$ and all $\lambda \in [0, 1]$. A mapping $f : X \to \mathbb{R}^m$ with $X \subseteq \mathbb{R}^n$ convex is called *convex* (resp. *concave*) iff

$$f(\lambda x + (1-\lambda)y) \leq (\text{resp. } \geq)\ \lambda f(x) + (1-\lambda) f(y) \tag{9}$$

for all $x, y \in X$ and all $\lambda \in [0, 1]$ (cf. e.g. Ortega and Rheinboldt, 1970). Note that f is concave iff $-f$ is convex. Note also that f is convex (resp. concave) iff $f_{i\cdot}$ is convex (resp. concave) for all $i = 1, \ldots, m$.

We extend the notion of convexity/concavity from $\mathbb{R}^n \to \mathbb{R}$ to $\overline{\mathbb{R}}^n \to \overline{\mathbb{R}}$ as follows: Let $f : \overline{\mathbb{R}}^n \to \overline{\mathbb{R}}$, and $I : \{1, \ldots, n\} \to \{-\infty, \mathrm{id}, \infty\}$. Here, $-\infty$ denotes the function that assigns $-\infty$ to every argument, id denotes the identity function, and ∞ denotes the function that assigns ∞ to every argument. We define the mapping $f^{(I)} : \overline{\mathbb{R}}^n \to \overline{\mathbb{R}}$ by

$$f^{(I)}(x) := f(I(1)(x_{1\cdot}), \ldots, I(n)(x_{n\cdot})) \quad \text{for all } x \in \overline{\mathbb{R}}^n. \tag{10}$$

A mapping $f : \overline{\mathbb{R}}^n \to \overline{\mathbb{R}}$ is called *concave* iff the following conditions are fulfilled for all $I : \{1, \ldots, n\} \to \{-\infty, \mathrm{id}, \infty\}$:

(1) $\mathrm{fdom}(f^{(I)})$ is convex.
(2) $f^{(I)}|_{\mathrm{fdom}(f^{(I)})}$ is concave.
(3) If $\mathrm{fdom}(f^{(I)}) \neq \emptyset$, then $f^{(I)}(x) < \infty$ for all $x \in \mathbb{R}^n$.

We need to quantify over all mappings $I : \{1, \ldots, n\} \to \{-\infty, \mathrm{id}, \infty\}$ in order to express that the function f is concave in all of its arguments, even if we fix some of them to $-\infty$ or ∞. The monotone function $f : \overline{\mathbb{R}}^2 \to \overline{\mathbb{R}}$ defined by

$$f(x_1, x_2) := \begin{cases} 0 & \text{if } x_1 < \infty \text{ or } x_2 < 0 \\ x_2^2 & \text{if } x_1 = \infty \text{ and } x_2 \geq 0 \end{cases} \qquad \text{for all } x_1, x_2 \in \overline{\mathbb{R}}, \tag{11}$$

for instance, is affine and thus concave on $\mathbb{R}^2$. However, according to the above definition, it is not concave on $\overline{\mathbb{R}}^2$, since $f(\infty, \cdot)$ is not concave (because $f(\infty, x_2) = x_2^2$ for all $x_2 \in \overline{\mathbb{R}}_{\geq 0}$). Fig. 3 shows the graph of a function $f : \overline{\mathbb{R}}^2 \to \overline{\mathbb{R}}$ that is monotone and concave.

A mapping $f : \overline{\mathbb{R}}^n \to \overline{\mathbb{R}}^m$ is called *concave* iff $f_{i\cdot}$ is concave for all $i \in \{1, \ldots, m\}$. A mapping $f : \overline{\mathbb{R}}^n \to \overline{\mathbb{R}}^m$ is called *convex* iff $-f$ is concave.

A function $f : \overline{\mathbb{R}}^n \to \overline{\mathbb{R}}^m$ is called *mcave* iff it is monotone and concave. It is called *cmcave* iff it is a mcave function and $f_{i\cdot}^{(I)}$ is upward-chain-continuous on $\{x \in \overline{\mathbb{R}}^n \mid f_{i\cdot}^{(I)}(x) > -\infty\}$ for all $I : \{1, \ldots, n\} \to \{-\infty, \mathrm{id}, \infty\}$ and all $i \in \{1, \ldots, n\}$.

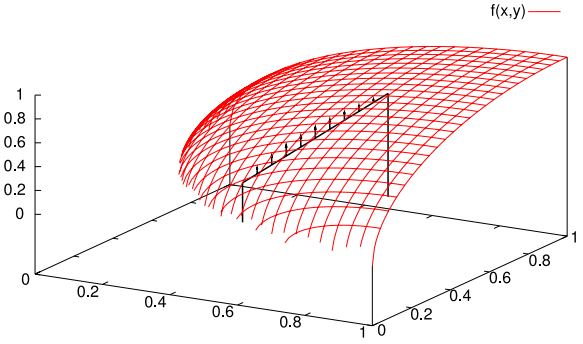


Fig. 3. Plot of a monotone and concave function $f : \overline{\mathbb{R}}^2 \to \overline{\mathbb{R}}$.

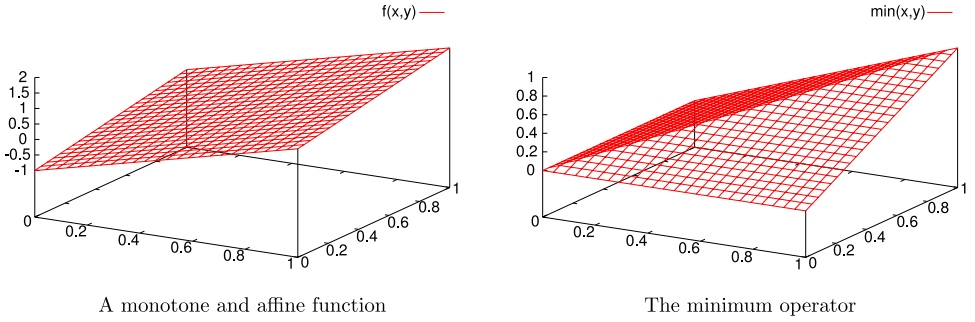


Fig. 4. Examples of monotone and concave functions.

A function $f : \mathbb{R}^n \to \mathbb{R}^m$ is called *affine* iff there exist some $A \in \mathbb{R}^{m\times n}$ and some $b \in \mathbb{R}^m$ such that $f(x) = Ax + b$ for all $x \in \mathbb{R}^n$. Accordingly, a function $f : \overline{\mathbb{R}}^n \to \overline{\mathbb{R}}^m$ is called *affine* iff there exist some $A \in \mathbb{R}^{m\times n}$ and some $b \in \overline{\mathbb{R}}^m$ such that $f(x) = Ax + b$ for all $x \in \overline{\mathbb{R}}^n$. Here, we use the convention that $-\infty + \infty = -\infty$. Observe that an affine function f with $f(x) = Ax + b$ is monotone, whenever all entries of A are non-negative.

Lemma 1. *Every affine function is convex, concave, and cmcave. The operator $\vee$ is convex, but not concave. The operator $\wedge$ is cmcave, but not convex (see Fig. 4).* □

3.3. Collecting semantics

In our programming model, we consider statements of the form

$$g(x) \leq 0;\ x := p(x) \tag{12}$$

where $x = (x_1, \ldots, x_n)^\top \in \mathbb{R}^n$ denotes the vector of program variables, and $g \in \mathbb{R}^k[x_1, \ldots, x_n]$ and $p \in \mathbb{R}^n[x_1, \ldots, x_n]$ are multivariate polynomials with coefficients from $\mathbb{R}^k$ and $\mathbb{R}^n$, respectively. Here, 0 also denotes the zero vector. An example is

$$x_1^2 + x_2^2 - 16 \leq 0;\ \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} := \frac{5}{4}\begin{pmatrix} x_2 \\ x_1 \end{pmatrix}. \tag{13}$$

It assigns $\frac{5}{4}$ of the value of the program variable x_i to the program variable x_{3-i} for all $i \in \{1, 2\}$, provided that $x_1^2 + x_2^2 - 16 \leq 0$ holds. A statement combines a guard with an assignment. The set of statements is denoted by **Stmt**. Statements of the form $g(x) \leq 0$, i.e., p is the identity function, are called *guards*. Statements of the form $x := p(x)$, i.e., $k = 0$, are called *assignments*. A statement $g(x) \leq 0;\ x := p(x)$ is called *affine* (resp. *quadratic*) iff the functions g and p are affine (resp. quadratic).

As usual in static program analysis by abstract interpretation, we refer to the program's collecting semantics, which safely over-approximates the concrete semantics. The *collecting semantics* $[\![s]\!] : 2^{\mathbb{R}^n} \to 2^{\mathbb{R}^n}$ of a statement $s \in \mathbf{Stmt}$ assigns a set $[\![s]\!]X$ of states after the execution of s to each set X of states before the execution of s. Here, a state of a program is modeled as a vector $x = (x_1, \ldots, x_n)^\top \in \mathbb{R}^n$. The collecting semantics of statements is defined by

$$[\![g(x) \leq 0;\ x := p(x)]\!]\, X := \{p(x) \mid x \in X,\ g(x) \leq 0\} \quad \text{for all } X \subseteq \mathbb{R}^n. \tag{14}$$

We represent programs by their control-flow graphs, i.e., a program G is a triple $(N, E, \mathbf{st})$, where N is a finite set of *program points*, $E \subseteq N \times \mathbf{Stmt} \times N$ is a finite set of control-flow edges, and $\mathbf{st} \in N$ is the start program point. As usual, the *collecting semantics* V of a program $G = (N, E, \mathbf{st})$ w.r.t. a set $I \subseteq \mathbb{R}^n$ of *initial states* is the least solution of the following constraint system:

$$\mathbf{V}[\mathbf{st}] \supseteq I \qquad \mathbf{V}[v] \supseteq [\![s]\!](\mathbf{V}[u]) \quad \text{for all } (u, s, v) \in E. \tag{15}$$

Here, the variables $\mathbf{V}[v]$, $v \in N$ take values in $2^{\mathbb{R}^n}$. The components of the collecting semantics V (i.e. the components of the least solution of (15)) are denoted by $V[v]$ for all $v \in N$. We use different fonts to distinguish between the variables of the constraint system and its least solution.

3.4. Abstract domain of polynomial templates

Next, we define the abstract domain we are going to use throughout this article. Following the lines of Adjé et al. (2010), we assume that we have given a fixed set

$$P \subseteq \mathbb{R}[x_1, \ldots, x_n] \tag{16}$$

of *polynomial templates* with coefficients from $\mathbb{R}$. P is called *linear* (resp. *quadratic*) iff all polynomials $p \in P$ are linear (resp. quadratic). Usually, P will consist of only *finitely many* templates.

Example 2 (*Adjé et al., 2010*)**.** The set $P = \{p_1, p_2, p_3, p_4, p_5\}$ with

$$\begin{aligned} p_1(x_1, x_2) &= -x_1 \qquad p_2(x_1, x_2) = x_1 \\ p_3(x_1, x_2) &= -x_2 \qquad p_4(x_1, x_2) = x_2 \\ p_5(x_1, x_2) &= 2x_1^2 + 3x_2^2 + 2x_1x_2 \end{aligned} \tag{17}$$

is a *set of polynomial templates*. More precisely, it is a finite set of *quadratic templates*. This set of quadratic templates is used to analyze the harmonic oscillator discussed in Section 2. □

Within this article, an abstract value is a mapping $\beta : P \to \overline{\mathbb{R}}$ that assigns an upper bound $\beta(q)$ to every polynomial $q \in P$. The abstract value β represents the set of all program states $x \in \mathbb{R}^n$ such that $q(x) \leq \beta(q)$ holds for all $q \in P$. Abstract values are partially ordered by the point-wise extension of $\leq$ which for simplicity is again denoted by $\leq$. That is, $\beta \leq \beta'$ if and only if $\beta(q) \leq \beta'(q)$ for all $q \in P$. Together with this partial ordering $P \to \overline{\mathbb{R}}$ forms a complete lattice.

Following the approach of abstract interpretation, we define a Galois-connection that consists of the *abstraction* $\alpha : 2^{\mathbb{R}^n} \to P \to \overline{\mathbb{R}}$ and the *concretization* $\gamma : (P \to \overline{\mathbb{R}}) \to 2^{\mathbb{R}^n}$ as follows:

$$\gamma(\beta) := \{x \in \mathbb{R}^n \mid \forall p \in P\ .\ p(x) \leq \beta(p)\} \quad \text{for all } \beta : P \to \overline{\mathbb{R}} \tag{18}$$

$$\alpha(X) := \bigwedge \{\beta : P \to \overline{\mathbb{R}} \mid \gamma(\beta) \supseteq X\} \quad \text{for all } X \subseteq \mathbb{R}^n. \tag{19}$$

As shown by Adjé et al. (2010), α and γ form a Galois-connection. The elements from $\gamma(P \to \overline{\mathbb{R}})$ and the elements from $\alpha(2^{\mathbb{R}^n})$ are called *closed*. $\alpha(\gamma(\beta))$ is called the *closure* of the abstract value $\beta : P \to \mathbb{R}$. Accordingly, $\gamma(\alpha(X))$ is called the *closure* of the set $X \subseteq \mathbb{R}^n$ of states. It is the minimal set of states that subsumes X and can be represented by an abstract value β.

Before we go further, we discuss some aspects of the closure operation $\alpha \circ \gamma$. For all abstract values $\beta : P \to \overline{\mathbb{R}}$ and all polynomial templates $r \in P$, we have

$$\alpha(\gamma(\beta))(r) = \sup\{r(x) \mid x \in \gamma(\beta)\} \tag{20}$$
$$= \sup\{r(x) \mid x \in \mathbb{R}^n \text{ and } \forall q \in P \,.\, q(x) \leq \beta(q)\} \tag{21}$$
$$= \inf\{-r(x) \mid x \in \mathbb{R}^n \text{ and } \forall q \in P \,.\, q(x) \leq \beta(q)\}. \tag{22}$$

The above equalities (cf. Adjé et al., 2010) lead to the following remarks:

Remark 3. If P is finite and all polynomial templates $q \in P$ with $\beta(q) < \infty$ (i.e., all polynomial templates that are bounded) are linear and r is quadratic (not necessarily concave), then $\alpha(\gamma(\beta))(r)$ can be computed by solving a *quadratic optimization problem* (cf. (22)). Solving quadratic optimization problems is **NP**-complete (see e.g. Vavasis, 1990). Vice versa, solving quadratic optimization problems is polynomial-time reducible to computing closures. Thus, computing closures is **NP**-hard.

Remark 4. If P is finite and linear, then closures can be computed by solving linear programming problems, i.e., in polynomial time.

Remark 5. If P is finite and all polynomial templates $q \in P$ with $\beta(q) < \infty$ (i.e., all polynomial templates that are bounded) are convex and r is concave (i.e. $-r$ is convex), then $\alpha(\gamma(\beta))(r)$ can be computed by solving a *convex optimization problem* (cf. (22)). If all polynomial templates $q \in P$ with $\beta(q) < \infty$ and r are additionally quadratic, then $\alpha(\gamma(\beta))(r)$ can be computed by solving a *convex quadratic optimization problem*. Convex quadratic optimization problems can be computed through semi-definite programming (see e.g. Todd, 2001).

Example 6 (*Adjé et al., 2010*)**.** We continue Example 2. Let

$$\beta = \{p_1 \mapsto 0,\ p_2 \mapsto 1,\ p_3 \mapsto 0,\ p_4 \mapsto 1,\ p_5 \mapsto \infty\}. \tag{23}$$

Then $\gamma(\beta) = [0, 1] \times [0, 1]$. The closure of β is

$$\alpha(\gamma(\beta)) = \{p_1 \mapsto 0,\ p_2 \mapsto 1,\ p_3 \mapsto 0,\ p_4 \mapsto 1,\ p_5 \mapsto 7\}, \tag{24}$$

because $\alpha(\gamma(\beta))(p_5) = \sup\{p_5(x_1, x_2) \mid (x_1, x_2)^\top \in \gamma(\beta)\} = 7$. □

3.5. Abstract semantics

The next step in static program analysis by abstract interpretation, is to abstract the operations used by the collecting semantics that operate on sets of concrete states, to abstract operations that operate on abstract values. Given the Galois connection introduced in Section 3.4, the *abstract semantics* $[\![s]\!]^\sharp : (P \to \overline{\mathbb{R}}) \to (P \to \overline{\mathbb{R}})$ of a statement s is defined by $[\![s]\!]^\sharp := \alpha \circ [\![s]\!] \circ \gamma$. Accordingly, the *abstract semantics* $V^\sharp$ of a program $G = (N, E, \mathbf{st})$ w.r.t. to a set $I \subseteq \mathbb{R}^n$ of initial states is the least solution of the following constraint system:

$$\mathbf{V}^\sharp[\mathbf{st}] \geq \alpha(I) \qquad \mathbf{V}^\sharp[v] \geq [\![s]\!]^\sharp(\mathbf{V}^\sharp[u]) \quad \text{for all } (u, s, v) \in E. \tag{25}$$

Here, the variables $\mathbf{V}^\sharp[v]$, $v \in N$ take values in $P \to \overline{\mathbb{R}}$. The components of the abstract semantics $V^\sharp$ are denoted by $V^\sharp[v]$ for all $v \in N$. The abstract semantics safely over-approximates the collecting semantics:

Lemma 7. *$V[v] \subseteq \gamma(V^\sharp[v])$ and $\alpha(V[v]) \leq V^\sharp[v]$ for all program points v.* □

In this article, we aim at using convex optimization to approximate the abstract semantics as precisely as possible. For that, as we will see later, it would be preferable if $[\![s]\!]^\sharp$ were concave (i.e., $-[\![s]\!]^\sharp$ were convex) for every statement s. Unfortunately, this property is not always fulfilled as indicated by the following example:

Example 8. Assume that $P = \{p_1, p_2, p_3\} \subseteq \mathbb{R}[x_1]$ with

$$p_1(x_1) = x_1 \qquad p_2(x_1) = -x_1 \qquad p_3(x_1) = x_1^2 \tag{26}$$

for all $x_1 \in \mathbb{R}$. We consider the statement $s = x_1 := x_1$, i.e., the statement s does not modify the state. Then, for all $\beta_x = \{p_1 \mapsto x, p_2 \mapsto 0, p_3 \mapsto \infty\}$ with $x \in \overline{\mathbb{R}}$, we have

$$(\llbracket s \rrbracket^\sharp \beta_x)(p_3) = \sup\{p_3(x_1) \mid x_1 \in \gamma(\beta_x)\} \tag{27}$$

$$= \sup\{x_1^2 \mid x_1 \in \mathbb{R} \text{ and } 0 \leq x_1 \leq x\} \tag{28}$$

$$= \begin{cases} x^2 & \text{if } x \geq 0 \\ -\infty & \text{if } x < 0. \end{cases} \tag{29}$$

Hence, we get

$$\left(\llbracket s \rrbracket^\sharp \left(\frac{1}{2}\beta_0 + \frac{1}{2}\beta_2\right)\right)(p_3) = (\llbracket s \rrbracket^\sharp(\beta_1))(p_3) = 1 \tag{30}$$

$$\not\geq 2 = \frac{1}{2}0 + \frac{1}{2}4 = \frac{1}{2}(\llbracket s \rrbracket^\sharp \beta_0)(p_3) + \frac{1}{2}(\llbracket s \rrbracket^\sharp \beta_2)(p_3). \tag{31}$$

This implies that $\llbracket s \rrbracket^\sharp$ is not concave. □

Nonetheless, in some cases $\llbracket s \rrbracket^\sharp$ is indeed concave. One important case is when all polynomials $q \in P$ are affine, and the statement s is affine. This case is studied by Costan et al. (2005) and by Gawlitza and Seidl (2007b).

Adjé et al. (2010) propose to use a convex relaxation of $-\llbracket s \rrbracket^\sharp$ (resp. concave relaxation of $\llbracket s \rrbracket^\sharp$) instead of $-\llbracket s \rrbracket^\sharp$ (resp. $\llbracket s \rrbracket^\sharp$). By doing so, an intractable NP-hard problem is approximated by a convex optimization problem. Convexity here has the advantage that a wide class of convex optimization problems can be solved efficiently. In the remainder, we will be faced with *semi-definite programming* problems which are special instances of convex optimization problems for which efficient interior point methods exist.

3.6. Relaxed abstract semantics

Adjé et al. (2010) propose to use *convex relaxation schemas* in order to approximate the abstract semantics. The abstract semantics $\llbracket s \rrbracket^\sharp$ of a statement s is replaced with a *relaxed abstract semantics* $\llbracket s \rrbracket^{\mathcal{R}}$ that fulfills the following properties:

(1) $\llbracket s \rrbracket^{\mathcal{R}} \geq \llbracket s \rrbracket^\sharp$, i.e., the relaxed abstract semantics $\llbracket s \rrbracket^{\mathcal{R}}$ of s safely over-approximates the abstract semantics $\llbracket s \rrbracket^\sharp$ of s.
(2) $\llbracket s \rrbracket^{\mathcal{R}}$ is cmcave.

The *relaxed abstract semantics* $V^{\mathcal{R}}$ of a program $G = (N, E, \mathbf{st})$ with initial states I is then defined as the least solution of the following constraint system over $P \to \overline{\mathbb{R}}$:

$$\mathbf{V}^{\mathcal{R}}[\mathbf{st}] \geq \alpha(I) \qquad \mathbf{V}^{\mathcal{R}}[v] \geq \llbracket s \rrbracket^{\mathcal{R}}(\mathbf{V}^{\mathcal{R}}[u]) \quad \text{for all } (u, s, v) \in E. \tag{32}$$

Here, the variables $\mathbf{V}^{\mathcal{R}}[v]$, $v \in N$ take values in $P \to \overline{\mathbb{R}}$. The components of the relaxed abstract semantics $V^{\mathcal{R}}$ are denoted by $V^{\mathcal{R}}[v]$ for all $v \in N$. The relaxed abstract semantics safely over-approximates the abstract semantics, and thus finally the collecting semantics and the concrete semantics:

Lemma 9. *$V^\sharp[v] \leq V^{\mathcal{R}}[v]$ for all program points v.* □

We emphasize that the set of all solutions of the constraints system (32), which defines the relaxed abstract semantics $V^{\mathcal{R}}$, is *not* always convex, although the relaxed abstract semantics $\llbracket s \rrbracket^{\mathcal{R}}$ is cmcave for each statement s. In consequence it is not possible to compute $V^{\mathcal{R}}$ directly through convex optimization techniques.

3.7. Obtaining a relaxed abstract semantics through semi-definite relaxation

In this subsection we briefly discuss the relaxed abstract semantics introduced by Adjé et al. (2010). This relaxed abstract semantics is based on Shor's semi-definite relaxation schema. This subsection is more technical than the remainder of this article. It is not essential, though, for the understanding of the remainder of this article. The purpose of this subsection is to demonstrate that a non-trivial relaxed abstract semantics exists that fulfills the requirements mentioned in Section 3.6.

3.7.1. Semi-definite programming

Let us now briefly introduce semi-definite programming. For more details we refer to, e.g. Boyd and Vandenberghe (2004) and Nemirovski (2005). Let $S\mathbb{R}^{n\times n}$ (resp. $S\mathbb{R}^{n\times n}_{+}$, resp. $S\mathbb{R}^{n\times n}_{++}$) denote the set of symmetric matrices (resp. the set of positive semi-definite matrices, resp. the set of positive definite matrices). A square matrix $A \in \mathbb{R}^{n\times n}$ is called *symmetric* iff $A^\top = A$. A symmetric matrix A is called positive semi-definite (resp. positive definite) iff $x^\top Ax \geq 0$ (resp. $x^\top Ax > 0$) for all $x \in \mathbb{R}^n$. We denote the Löwner ordering of symmetric matrices by $\preceq$, i.e., $A \preceq B$ iff $B - A \in S\mathbb{R}^{n\times n}_{+}$. We write $A \prec B$ iff $B - A \in S\mathbb{R}^{n\times n}_{++}$. $\mathrm{Tr}(A)$ denotes the trace of a square matrix $A \in \mathbb{R}^{n\times n}$, i.e., $\mathrm{Tr}(A) = \sum_{i=1}^{n} A_{i\cdot i}$. The inner product of two matrices A and B is denoted by $A \bullet B$, i.e., $A \bullet B = \mathrm{Tr}(A^\top B)$. For $\mathcal{A} = (A_1, \ldots, A_m)$ with $A_i \in \mathbb{R}^{n\times n}$ for all $i = 1, \ldots, m$, we denote the vector $(A_1 \bullet X, \ldots, A_m \bullet X)^\top$ by $\mathcal{A}(X)$. We consider semi-definite programming problems (SDP problems for short) of the form

$$z^* = \sup \{C \bullet X \mid X \in S\mathbb{R}^{n\times n}_{+},\ \mathcal{A}(X) = a,\ \mathcal{B}(X) \leq b\}, \tag{33}$$

where $\mathcal{A} = (A_1, \ldots, A_m)$, $A_1, \ldots, A_m \in S\mathbb{R}^{n\times n}$, $a \in \mathbb{R}^m$, $\mathcal{B} = (B_1, \ldots, B_k)$, $B_1, \ldots, B_k \in S\mathbb{R}^{n\times n}$, $b \in \mathbb{R}^k$, and $C \in S\mathbb{R}^{n\times n}$. The value $z^* \in \overline{\mathbb{R}}$ is called the *optimal value*. The set $\{X \in S\mathbb{R}^{n\times n}_{+} \mid \mathcal{A}(X) = a,\ \mathcal{B}(X) \leq b\}$ is called the *feasible space*. An element of the feasible space, is called *feasible solution*. The problem is called *infeasible* iff the feasible space is empty, i.e., $z^* = -\infty$. It is called *unbounded* iff $z^* = \infty$. A feasible solution X^* is called *optimal solution* iff $C \bullet X^* = z^*$.

3.7.2. The relaxation

For the remainder of this subsection we assume that P is finite, all templates $p \in P$ are quadratic (but not necessarily convex), and all statements are of the form $g(x) \leq 0;\ x := p(x)$, where g is quadratic and p is affine. The goal is to define a relaxed abstract semantics which satisfies the properties described in Section 3.6. For that, we use Shor's semi-definite relaxation schema.

Let $s = g(x) \leq 0;\ x := p(x)$ be a statement. Recall that the abstract semantics $[\![s]\!]^\sharp$ of s is given by

$$([\![s]\!]^\sharp\beta)(r) = \sup \left\{r(p(x)) \mid x \in \mathbb{R}^n \text{ and } g(x) \leq 0 \text{ and } \forall q \in P\ .\ q(x) \leq \beta(q)\right\} \tag{34}$$

for all abstract values $\beta : P \to \overline{\mathbb{R}}$ and all templates $r \in P$. Because g is quadratic and p is affine, we had to solve a non-linear optimization problem for computing $([\![s]\!]^\sharp\beta)(r)$. Unfortunately, this non-linear optimization problem is not necessarily convex. Using (the dual version of) Shor's semi-definite relaxation schema, we relax the abstract semantics $[\![s]\!]^\sharp$ of s as follows. W.l.o.g., we assume:

(1) For every polynomial $q \in \mathbb{R}[x_1, \ldots, x_n]$ with coefficients from $\mathbb{R}$, there are some $A_q \in S\mathbb{R}^{n\times n}$, some $b_q \in \mathbb{R}^n$, and some $c_q \in \mathbb{R}$ such that

$$q(x) = x^\top A_q x + 2{b_q}^\top x + c_q. \tag{35}$$

(2) $p(x) = Ax + b$ with $A \in \mathbb{R}^{n\times n}$ and $b \in \mathbb{R}^n$.

For all $\beta : P \to \mathbb{R}$, and all $r \in P$, we then get

$$
\begin{aligned}
([\![s]\!]^\sharp\beta)(r) &= \sup\ \left\{ r(p(x)) \mid x \in \mathbb{R}^n, g(x) \leq 0,\ \forall q \in P\ .\ q(x) \leq \beta(q) \right\} \\
&= \sup \Bigg\{ r(Ax+b) \mid x \in \mathbb{R}^n, \\
&\quad \forall i \in \{1,\dots,k\}\ .\ x^\top A_{g_i\cdot} x + 2b_{g_i\cdot}^\top x + c_{g_i\cdot} \leq 0, \\
&\quad \forall q \in P\ .\ x^\top A_q x + 2b_q^\top x + c_q \leq \beta(q) \Bigg\} \\
&= \sup \Bigg\{ x^\top A^\top A_r A x + 2b^\top A_r A x + 2b_r^\top A x + b^\top A_r b + 2b_r^\top b + c_r \mid \\
&\quad x \in \mathbb{R}^n, \\
&\quad \forall i \in \{1,\dots,k\}\ .\ x^\top A_{g_i\cdot} x + 2b_{g_i\cdot}^\top x + c_{g_i\cdot} \leq 0, \\
&\quad \forall q \in P\ .\ x^\top A_q x + 2b_q^\top x + c_q \leq \beta(q) \Bigg\} \\
&= \sup \{ (1, x^\top) \begin{pmatrix} b^\top A_r b + 2b_r^\top b + c_r & b^\top A_r A + b_r^\top A \\ (b^\top A_r A + b_r^\top A)^\top & A^\top A_r A \end{pmatrix} \begin{pmatrix} 1 \\ x \end{pmatrix} \mid \\
&\quad x \in \mathbb{R}^n, \\
&\quad \forall i \in \{1,\dots,k\}\ .\ (1, x^\top) \begin{pmatrix} c_{g_i\cdot} & b_{g_i\cdot}^\top \\ b_{g_i\cdot} & A_{g_i\cdot} \end{pmatrix} \begin{pmatrix} 1 \\ x \end{pmatrix} \leq 0, \\
&\quad \forall q \in P\ .\ (1, x^\top) \begin{pmatrix} c_q & b_q{}^\top \\ b_q & A_q \end{pmatrix} \begin{pmatrix} 1 \\ x \end{pmatrix} \leq \beta(q) \} \\
&= \sup \{ \begin{pmatrix} b^\top A_r b + 2b_r^\top b + c_r & b^\top A_r A + b_r^\top A \\ (b^\top A_r A + b_r^\top A)^\top & A^\top A_r A \end{pmatrix} \bullet \begin{pmatrix} 1 \\ x \end{pmatrix} (1, x^\top) \mid \\
&\quad x \in \mathbb{R}^n, \\
&\quad \forall i \in \{1,\dots,k\}\ .\ \begin{pmatrix} c_{g_i\cdot} & b_{g_i\cdot}^\top \\ b_{g_i\cdot} & A_{g_i\cdot} \end{pmatrix} \bullet \begin{pmatrix} 1 \\ x \end{pmatrix} (1, x^\top) \leq 0, \\
&\quad \forall q \in P\ .\ \begin{pmatrix} c_q & b_q{}^\top \\ b_q & A_q \end{pmatrix} \bullet \begin{pmatrix} 1 \\ x \end{pmatrix} (1, x^\top) \leq \beta(q) \} \\
&\leq \sup \Bigg\{ \begin{pmatrix} b^\top A_r b + 2b_r^\top b + c_r & b^\top A_r A + b_r^\top A \\ (b^\top A_r A + b_r^\top A)^\top & A^\top A_r A \end{pmatrix} \bullet X \mid \\
&\quad X \succeq 0,\ X_{1\cdot 1} = 1 \\
&\quad \forall i \in \{1,\dots,k\}\ .\ \begin{pmatrix} c_{g_i\cdot} & b_{g_i\cdot}^\top \\ b_{g_i\cdot} & A_{g_i\cdot} \end{pmatrix} \bullet X \leq 0, \\
&\quad \forall q \in P\ .\ \begin{pmatrix} c_q & b_q{}^\top \\ b_q & A_q \end{pmatrix} \bullet X \leq \beta(q) \Bigg\}.
\end{aligned}
$$

The last inequality holds, because $X \succeq 0$ and $X_{1\cdot 1} = 1$ hold for all X and all x with

$$X = \begin{pmatrix} 1 \\ x \end{pmatrix} (1, x^\top). \tag{36}$$

Because of the above inequality, we define the relaxed abstract semantics $[\![s]\!]^{\mathcal{R}}$ of s by

$$\begin{aligned}([\![s]\!]^{\mathcal{R}}\beta)(r) := \sup \Bigg\{ & \begin{pmatrix} b^\top A_r b + 2b_r^\top b + c_r & b^\top A_r A + b_r^\top A \\ (b^\top A_r A + b_r^\top A)^\top & A^\top A_r A \end{pmatrix} \bullet X \mid \\ & X \succeq 0,\ X_{1\cdot 1} = 1 \\ & \forall i \in \{1,\dots,k\} .\ \begin{pmatrix} c_{g_i\cdot} & b_{g_i\cdot}^\top \\ b_{g_i\cdot} & A_{g_i\cdot} \end{pmatrix} \bullet X \leq 0, \\ & \forall q \in P .\ \begin{pmatrix} c_q & b_q^\top \\ b_q & A_q \end{pmatrix} \bullet X \leq \beta(q) \Bigg\}. \end{aligned} \tag{37}$$

The important properties of the relaxed abstract semantics $[\![s]\!]^{\mathcal{R}}$ are summarized in the following lemma:

Lemma 10 (*Adjé et al., 2010; Gawlitza and Seidl, 2010*)**.** *Let $s = g(x) \leq 0;\ x := p(x)$ be a statement, where g is quadratic and p is affine. Assume that P is finite and all $q \in P$ are quadratic. For the relaxed abstract semantics $[\![s]\!]^{\mathcal{R}}$ of s as defined in (37) we have:*

(1) $[\![s]\!]^{\mathcal{R}} \geq [\![s]\!]^{\sharp}$.
(2) $[\![s]\!]^{\mathcal{R}}$ *is cmcave.*
(3) $([\![s]\!]^{\mathcal{R}}\beta)(r) = ([\![s]\!]^{\sharp}\beta)(r)$, *whenever r is concave, g is convex, and all polynomial templates $q \in P$ with $\beta(q) < \infty$ are convex. This is in particular the case, whenever s is affine and all polynomial templates $q \in P$ are affine.* □

Because of the last statement of the above lemma, the methods to be presented here can be considered as a generalization of the methods developed by Gaubert et al. (2007) and Gawlitza and Seidl (2007b).

3.8. Systems of inequations over $\overline{\mathbb{R}}$

We want to reduce the problem of computing the relaxed abstract semantics $V^{\mathcal{R}}$ of a program G w.r.t. a set $I \subseteq \mathbb{R}^n$ of initial states to solving a system $\mathcal{C}(G, I)$ of inequalities of the form $\mathbf{x} \geq e$ over $\overline{\mathbb{R}}$, where each right-hand side e is cmcave. We set up this system $\mathcal{C}(G, I)$ as follows:

$$\mathbf{x}_{\mathbf{st},p} \geq \alpha(I)(p) \quad \text{for all } p \in P \tag{38}$$

$$\mathbf{x}_{v,p} \geq ([\![s]\!]^{\mathcal{R}}\{q \mapsto \mathbf{x}_{u,q} \mid q \in P\})(p) \quad \text{for all } (u, s, v) \in E, \text{ and all } p \in P. \tag{39}$$

The system $\mathcal{C}(G, I)$ of inequalities uses the set $\mathbf{X} = \{\mathbf{x}_{v,p} \mid v \in N \text{ and } p \in P\}$ of variables. For every $v \in N$ and every $p \in P$, The variable $\mathbf{x}_{v,p}$ receives the value for the upper bound on the polynomial template p at program point v. The relaxed abstract semantics of G w.r.t. to the set I of initial states can finally be read off the least solution of the system $\mathcal{C}(G, I)$ of inequalities over $\overline{\mathbb{R}}$:

Lemma 11. *Let $\rho^* : \mathbf{X} \to \overline{\mathbb{R}}$ denote the least solution of the system $\mathcal{C}(G, I)$ of inequalities. Then $V^{\mathcal{R}}[v](p) = \rho^*(\mathbf{x}_{v,p})$ for all $v \in N$ and all $p \in P$.* □

Because of the above lemma, it remains to provide methods for approximating or computing the least solution of $\mathcal{C}(G, I)$. This is the topic of the next sections.

Example 12. We continue our running example from Section 2. The set $P = \{p_1, \dots, p_5\} \subseteq \mathbb{R}[x_1, x_2]$ of quadratic templates we consider for this example is given by

$$p_1(x_1, x_2) = -x_1 \qquad p_2(x_1, x_2) = x_1 \qquad p_3(x_1, x_2) = -x_2 \tag{40}$$

$$p_4(x_1, x_2) = x_2 \qquad p_5(x_1, x_2) = 2x_1^2 + 3x_2^2 + 2x_1x_2 \tag{41}$$

for all $x_1, x_2 \in \mathbb{R}$. By Lemma 11, the relaxed abstract semantics is given by the least solution of the following system of inequalities:

$$\mathbf{x}_{\mathbf{st},p_1} \geq 0 \qquad \mathbf{x}_{\mathbf{st},p_1} \geq ([\![s]\!]^{\mathcal{R}}\{p \mapsto \mathbf{x}_{\mathbf{st},p} \mid p \in P\})(p_1) \tag{42}$$
$$\mathbf{x}_{\mathbf{st},p_2} \geq 1 \qquad \mathbf{x}_{\mathbf{st},p_2} \geq ([\![s]\!]^{\mathcal{R}}\{p \mapsto \mathbf{x}_{\mathbf{st},p} \mid p \in P\})(p_2) \tag{43}$$
$$\mathbf{x}_{\mathbf{st},p_3} \geq 0 \qquad \mathbf{x}_{\mathbf{st},p_3} \geq ([\![s]\!]^{\mathcal{R}}\{p \mapsto \mathbf{x}_{\mathbf{st},p} \mid p \in P\})(p_3) \tag{44}$$
$$\mathbf{x}_{\mathbf{st},p_4} \geq 1 \qquad \mathbf{x}_{\mathbf{st},p_4} \geq ([\![s]\!]^{\mathcal{R}}\{p \mapsto \mathbf{x}_{\mathbf{st},p} \mid p \in P\})(p_4) \tag{45}$$
$$\mathbf{x}_{\mathbf{st},p_5} \geq 7 \qquad \mathbf{x}_{\mathbf{st},p_5} \geq ([\![s]\!]^{\mathcal{R}}\{p \mapsto \mathbf{x}_{\mathbf{st},p} \mid p \in P\})(p_5). \tag{46}$$

Here,

$$s = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} := \begin{pmatrix} 1 & 0.01 \\ -0.01 & 0.99 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

and, according to equality (37),

$$\begin{aligned} &([\![s]\!]^{\mathcal{R}}\{p \mapsto \mathbf{x}_{\mathbf{st},p} \mid p \in P\})(p_i) \\ &\quad = \sup\{C_i \bullet X \mid X \succeq 0, X_{1\cdot 1} = 1, B_1 \bullet X \leq \mathbf{x}_{\mathbf{st},p_1}, \cdots, B_5 \bullet X \leq \mathbf{x}_{\mathbf{st},p_5}\} \end{aligned}$$

for all $i \in \{1, \ldots, 5\}$, where

$$B_1 = \begin{pmatrix} 0 & -0.5 & 0 \\ -0.5 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad B_2 = \begin{pmatrix} 0 & 0.5 & 0 \\ 0.5 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$
$$B_3 = \begin{pmatrix} 0 & 0 & -0.5 \\ 0 & 0 & 0 \\ -0.5 & 0 & 0 \end{pmatrix} \quad B_4 = \begin{pmatrix} 0 & 0 & 0.5 \\ 0 & 0 & 0 \\ 0.5 & 0 & 0 \end{pmatrix}$$
$$B_5 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 2 & 1 \\ 0 & 1 & 3 \end{pmatrix}$$

$$C_1 = \begin{pmatrix} 0 & -0.5 & -0.005 \\ -0.5 & 0 & 0 \\ -0.005 & 0 & 0 \end{pmatrix} \quad C_2 = \begin{pmatrix} 0 & 0.5 & 0.005 \\ 0.5 & 0 & 0 \\ 0.005 & 0 & 0 \end{pmatrix}$$
$$C_3 = \begin{pmatrix} 0 & 0.005 & -0.495 \\ 0.005 & 0 & 0 \\ -0.495 & 0 & 0 \end{pmatrix} \quad C_4 = \begin{pmatrix} 0 & -0.005 & 0.495 \\ -0.005 & 0 & 0 \\ 0.495 & 0 & 0 \end{pmatrix}$$
$$C_5 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1.9803 & 0.9802 \\ 0 & 0.9802 & 2.9603 \end{pmatrix}.$$

The above system of inequalities has the same least solution as the following system of equations:

$$\begin{aligned} \mathbf{x}_{\mathbf{st},p_1} &= 0 \vee ([\![s]\!]^{\mathcal{R}}\{p \mapsto \mathbf{x}_{\mathbf{st},p} \mid p \in P\})(p_1) \\ \mathbf{x}_{\mathbf{st},p_2} &= 1 \vee ([\![s]\!]^{\mathcal{R}}\{p \mapsto \mathbf{x}_{\mathbf{st},p} \mid p \in P\})(p_2) \\ \mathbf{x}_{\mathbf{st},p_3} &= 0 \vee ([\![s]\!]^{\mathcal{R}}\{p \mapsto \mathbf{x}_{\mathbf{st},p} \mid p \in P\})(p_3) \\ \mathbf{x}_{\mathbf{st},p_4} &= 1 \vee ([\![s]\!]^{\mathcal{R}}\{p \mapsto \mathbf{x}_{\mathbf{st},p} \mid p \in P\})(p_4) \\ \mathbf{x}_{\mathbf{st},p_5} &= 7 \vee ([\![s]\!]^{\mathcal{R}}\{p \mapsto \mathbf{x}_{\mathbf{st},p} \mid p \in P\})(p_5). \end{aligned} \tag{47}$$

In Section 5 we explain how to solve equation system (47) through the $\wedge$-strategy iteration approach. In Section 6 we will do this using the $\vee$-strategy iteration approach. $\square$

4. Systems of $\vee$-Cmcave equations

This section introduces the main object of our studies, namely *systems of $\vee$-cmcave equations*. How these equation systems can be solved through strategy iteration will be explained in the following sections.

Assume that a fixed set $\mathbf{X}$ of variables and a domain $\mathbb{D}$ is given. We consider equations of the form $\mathbf{x} = e$, where $\mathbf{x} \in \mathbf{X}$ is a variable and e is an expression over $\mathbb{D}$. A *system* $\mathcal{E}$ of equations is a finite set

$$\mathcal{E} = \{\mathbf{x}_1 = e_1, \ldots, \mathbf{x}_n = e_n\}$$

of equations, where $\mathbf{x}_1, \ldots, \mathbf{x}_n$ are pairwise distinct variables. We denote the set $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ of variables occurring in $\mathcal{E}$ by $\mathbf{X}_\mathcal{E}$. We drop the subscript, whenever it is clear from the context.

For a variable assignment $\rho : \mathbf{X} \to \mathbb{D}$, an expression e is mapped to a value $[\![e]\!]\rho$ by setting $[\![\mathbf{x}]\!]\rho := \rho(\mathbf{x})$, and $[\![f(e_1, \ldots, e_k)]\!]\rho := f([\![e_1]\!]\rho, \ldots, [\![e_k]\!]\rho)$, where $\mathbf{x} \in \mathbf{X}$, f is a k-ary operator ($k = 0$ is possible; then f is a constant), for instance $+$, and $e_1, \ldots, e_k$ are expressions. Let $\mathcal{E}$ be a system of equations. We define the unary operator $[\![\mathcal{E}]\!]$ on $\mathbf{X} \to \mathbb{D}$ by setting $([\![\mathcal{E}]\!]\rho)(\mathbf{x}) := [\![e]\!]\rho$ for all $\mathbf{x} = e \in \mathcal{E}$. A solution is a variable assignment ρ such that $\rho = [\![\mathcal{E}]\!]\rho$ holds. The set of solutions is denoted by $\mathbf{Sol}(\mathcal{E})$.

Assume in the following that $\mathbb{D}$ is a complete lattice. An expression e (resp. an equation $\mathbf{x} = e$) is called *monotone* iff all operators occurring in e are monotone.

The set $\mathbf{X} \to \mathbb{D}$ of all *variable assignments* is a complete lattice. For $\rho, \rho' : \mathbf{X} \to \mathbb{D}$, we write $\rho \lhd \rho'$ (resp. $\rho \rhd \rho'$) iff $\rho(\mathbf{x}) < \rho'(\mathbf{x})$ (resp. $\rho(\mathbf{x}) > \rho'(\mathbf{x})$) for all $\mathbf{x} \in \mathbf{X}$. For $d \in \mathbb{D}$, $\underline{d}$ denotes the variable assignment $\{\mathbf{x} \mapsto d \mid \mathbf{x} \in \mathbf{X}\}$. A variable assignment ρ with $\underline{\bot} \lhd \rho \lhd \underline{\top}$ is called *finite*. A pre-solution (resp. post-solution) is a variable assignment ρ such that $\rho \leq [\![\mathcal{E}]\!]\rho$ (resp. $\rho \geq [\![\mathcal{E}]\!]\rho$) holds. The set of pre-solutions (resp. the set of post-solutions) is denoted by $\mathbf{PreSol}(\mathcal{E})$ (resp. $\mathbf{PostSol}(\mathcal{E})$). The least fixpoint (resp. the greatest fixpoint) of an operator $f : \mathbb{D} \to \mathbb{D}$ is denoted by μf (resp. νf), provided that it exists. Thus, the least solution (resp. the greatest solution) of a system $\mathcal{E}$ of equations is denoted by $\mu[\![\mathcal{E}]\!]$ (resp. $\nu[\![\mathcal{E}]\!]$), provided that it exists. For a pre-solution ρ (resp. for a post-solution ρ), $\mu_{\geq\rho}[\![\mathcal{E}]\!]$ (resp. $\nu_{\leq\rho}[\![\mathcal{E}]\!]$) denotes the least solution that is greater than or equal to ρ (resp. the greatest solution that is less than or equal to ρ). In our setting, Knaster–Tarski's fixpoint theorem can be stated as follows: Every system $\mathcal{E}$ of monotone equations over a complete lattice has a least solution $\mu[\![\mathcal{E}]\!]$ and a greatest solution $\nu[\![\mathcal{E}]\!]$. Furthermore, $\mu[\![\mathcal{E}]\!] = \bigwedge \mathbf{PostSol}(\mathcal{E})$ and $\nu[\![\mathcal{E}]\!] = \bigvee \mathbf{PreSol}(\mathcal{E})$.

$\vee$-Cmcave Equations. An expression e (resp. equation $\mathbf{x} = e$) over $\overline{\mathbb{R}}$ is called *cmcave expression* (resp. *cmcave equation*) iff $[\![e]\!]$ is cmcave. An expression e (resp. equation $\mathbf{x} = e$) over $\overline{\mathbb{R}}$ is called *$\vee$-cmcave* iff $e = e_1 \vee \cdots \vee e_k$, where $e_1, \ldots, e_k$ are cmcave expressions.

Example 13. The operator $\sqrt{\cdot} : \overline{\mathbb{R}} \to \overline{\mathbb{R}}$ (defined by $\sqrt{x} = \sup\{y \in \mathbb{R} \mid y^2 \leq x\}$ for all $x \in \overline{\mathbb{R}}$) is cmcave. The least solution of the system $\mathcal{E} = \{\mathbf{x} = \frac{1}{2} \vee \sqrt{\mathbf{x}}\}$ of $\vee$-cmcave equations is $\mu[\![\mathcal{E}]\!] = 1$. $\square$

5. The min-strategy iteration approach

In this section we present the $\wedge$-strategy iteration approach of Costan et al. (2005). The general framework is explained in Section 5.1. After that we specialize the general $\wedge$-strategy iteration algorithm to an algorithm for solving systems of $\vee$-cmcave equations as introduced in Section 4. For that, we first show how to compute least solutions of systems of inequalities of the form $\mathbf{x}_k \geq f(\mathbf{x}_1, \ldots, \mathbf{x}_n)$, where f is an affine operator on $\overline{\mathbb{R}}$. This algorithm will later be used for evaluating $\wedge$-strategies. Then, in Section 5.3, we answer the question how the set of $\wedge$-strategies that is defined by a system of $\vee$-cmcave equations looks like. In Section 5.4 we utilize convex optimization for computing an improvement of a $\wedge$-strategy. In Section 5.5 we apply the developed $\wedge$-strategy improvement algorithm to the harmonic oscillator discussed in Section 2.

5.1. The general framework

Let $\mathbb{D}$ be a complete lattice. We are interested in computing a small fixpoint of a monotone self-map $f : \mathbb{D} \to \mathbb{D}$, where we assume that $f(x) = \min\{\pi(x) \mid \pi \in \Pi\}$ for all $x \in \mathbb{D}$. Here, Π is a family of "simpler" self-maps on $\mathbb{D}$. Observe that, for all $x \in \mathbb{D}$, $f(x) = \min\{\pi(x) \mid \pi \in \Pi\}$ iff $f(x) = \bigwedge\{\pi(x) \mid \pi \in \Pi\}$ and there exists a $\pi \in \Pi$ such that $f(x) = \pi(x)$. The term "simpler" in practice means that we assume that for any $\pi \in \Pi$, the least fixpoint $\mu\pi$ of π can be computed efficiently. The self-maps $\pi \in \Pi$ are the $\wedge$-*strategies* for f. $\vee$-strategies can be defined dually. Then we assume that $f(x) = \max\{\sigma(x) \mid \sigma \in \Sigma\}$ for all $x \in \mathbb{D}$, where Σ is a family of "simpler" self-maps on $\mathbb{D}$.

Example 14. Consider the following system of $\vee$-cmcave equations:

$$\mathbf{x} = 0 \vee \left(\frac{1}{2} \cdot \mathbf{x} + 1 \wedge 10\right). \tag{48}$$

Let $f(\mathbf{x})$ denote the right-hand side, i.e., the function $f : \overline{\mathbb{R}} \to \overline{\mathbb{R}}$ is defined by $f(x) = 0 \vee \left(\frac{1}{2} \cdot x + 1 \wedge 10\right)$ for all $x \in \overline{\mathbb{R}}$. Observe that $f(x) = \min\{\pi_1(x), \pi_2(x)\}$ for all $x \in \overline{\mathbb{R}}$, where

$$\pi_1(x) = 0 \vee \frac{1}{2} \cdot x + 1, \quad \text{and} \quad \pi_2(x) = 0 \vee 10 = 10 \quad \text{for all } x \in \overline{\mathbb{R}}. \tag{49}$$

Hence, π_1 and π_2 are the $\wedge$-strategies for f. Moreover, $f(x) = \max\{\sigma_1(x), \sigma_2(x)\}$ for all $x \in \overline{\mathbb{R}}$, where

$$\sigma_1(x) = 0, \quad \text{and} \quad \sigma_2(x) = \frac{1}{2} \cdot x + 1 \wedge 10 \quad \text{for all } x \in \overline{\mathbb{R}}. \tag{50}$$

Hence, σ_1 and σ_2 are the $\vee$-strategies for f. □

Since $f(x) = \min\{\pi(x) \mid \pi \in \Pi\}$ for all $x \in \mathbb{D}$, we get

$$\mu f = \min\{\mu\pi \mid \pi \in \Pi\}. \tag{51}$$

Eq. (51) can be shown as follows: Since $\mu\pi$ is a post-fixpoint of f for all $\pi \in \Pi$, the fixpoint theorem of Knaster–Tarski gives us that $\mu f \leq \mu\pi$ for all $\pi \in \Pi$ (*). Moreover, there exists some $\pi^* \in \Pi$ such that $\pi^*(\mu f) = f(\mu f)$. Since μf is a fixpoint of f, it is thus also a fixpoint of π^*. Therefore, $\mu\pi^* \leq \mu f$. Together with (*) we get Eq. (51).

However, if $f(x) = \max\{\sigma(x) \mid \sigma \in \Sigma\}$ for all $x \in \mathbb{D}$, then we can only conclude that $\nu f = \max\{\nu\sigma \mid \sigma \in \Sigma\}$. This is the dual of (51). The statement $\mu f = \max\{\mu\sigma \mid \sigma \in \Sigma\}$ does not hold in general as the following example shows:

Example 15. We continue Example 14. We have

$$\max\{\mu\sigma_1, \mu\sigma_2\} = \max\{0, -\infty\} = 0 < \mu f = 2 = \min\{2, 10\} = \min\{\mu\pi_1, \mu\pi_2\}. \tag{52}$$

Hence, if we can compute the least fixpoints of the "simpler" self-maps π_1 and π_2, then we can compute the least fixpoint of the self-map f. However, being able to compute the least fixpoints of the "simpler" self-maps σ_1 and σ_2 does not help in computing the least fixpoint of f. □

If we assume that Π is finite and we can compute $\mu\pi$ for every $\pi \in \Pi$, we immediately obtain a method for computing μf. However, this does not necessarily lead to a practical algorithm, since the cardinality of Π may be large, for instance, exponential in the size of the input. For tackling this problem, the idea is to start with an arbitrary $\wedge$-strategy π_0 and improve this $\wedge$-strategy iteratively utilizing the assumption that $f(x) = \min\{\pi(x) \mid \pi \in \Pi\}$ for all $x \in \mathbb{D}$. This idea can be formalized

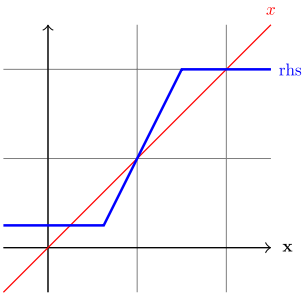


Fig. 5. The graphs of $\mathbf{x}$ and $(0.25 \vee 2 \cdot \mathbf{x} - 1) \wedge 2$.

as follows:

Algorithm 1 The $\wedge$-Strategy Improvement Algorithm

(1) *Initialization.* Set $k = 0$ and select any $\wedge$-strategy $\pi^{(0)} \in \Pi$.
(2) *Value determination.* Compute the least fixpoint $x^{(k)} := \mu\pi^{(k)}$ of $\pi^{(k)}$.
(3) If $x^{(k)} = f(x^{(k)})$, then return $x^{(k)}$.
(4) *$\wedge$-Strategy Improvement.* Take $\pi^{(k+1)} \in \Pi$ such that $f(x^{(k)}) = \pi^{(k+1)}(x^{(k)})$. Increment k by 1 and goto Step 2.

The fixpoint theorem of Knaster–Tarski gives us:

Theorem 16. (1) *$(x^{(i)})_{i\in\mathbb{N}}$ is a decreasing sequence of post-fixpoints of f (i.e., $x^{(i)} \geq f(x^{(i)})$ for all $i \in \mathbb{N}$) that is strictly decreasing until it is stable.*
(2) *If it is stable, then we have found a solution, i.e., a fixpoint of f.*
(3) *$x^{(i)}$ is greater than or equal to the least solution for all $i \in \mathbb{N}$, i.e., $x^{(i)} \geq \mu f$ for all $i \in \mathbb{N}$.*
(4) *The sequence $(x^{(i)})_i$ is bounded from above by the sequence obtained by Kleene iteration, i.e., $x^{(i)} \leq f^i(x^{(0)})$ for all $i \in \mathbb{N}$.*
(5) *If the set Π of all $\wedge$-strategies is finite, then termination is guaranteed after at most $|\Pi|$ steps.* □

Example 17. We continue with Example 14. Assume that $\pi^{(0)} = \pi_2$. Then we get $x^{(0)} = \mu\pi^{(0)} = \mu\pi_2 = 10$. We observe that $x^{(0)}$ is not a solution of f, because $x^{(0)} = 10 > 6 = f(10) = f(x^{(0)})$. Hence, we improve the current $\wedge$-strategy. For that we observe that

$$\pi_1(x^{(0)}) = \pi_1(10) = 6 = f(x^{(0)}) < 10 = \pi_2(10) = \pi_2(x^{(0)}). \tag{53}$$

Hence, the algorithm chooses $\pi^{(1)} = \pi_1$ as the next $\wedge$-strategy. Thus, we get $x^{(1)} = \mu\pi^{(1)} = \mu\pi_1 = 2$. As we will see in the following, we can use linear programming to compute $\mu\pi_1$. Since $f(x^{(1)}) = f(2) = 2 = x^{(1)}$, we have found a fixpoint of f. Hence, the algorithm terminates. We have found the least fixpoint μf of f. □

In the above example, we have found the least fixpoint μf of f. However, Algorithm 1 stops whenever *some* fixpoint $x^{(k)}$ is reached, not necessarily the least one. We give a simple example for this phenomenon:

Example 18. Consider the following system of $\vee$-cmcave equations:

$$\mathbf{x} = (0.25 \vee 2 \cdot \mathbf{x} - 1) \wedge 2. \tag{54}$$

The graph of the left-hand side $\mathbf{x}$ and the graph of the right-hand side $(0.25 \vee 2 \cdot \mathbf{x} - 1) \wedge 2$ are drawn in Fig. 5. The least solution of the above system of $\vee$-cmcave equations is $\mathbf{x} = 0.25$. The set $\Pi = \{\pi_1, \pi_2\}$ of $\wedge$-strategies for the right-hand side is given by $\pi_1(x) = (0.25 \vee 2 \cdot x - 1)$ and $\pi_2(x) = 2$ for all x. If we initialize the $\wedge$-strategy iteration with the $\wedge$-strategy $\pi^{(0)} = \pi_2$, the algorithm returns 2, since the $\wedge$-strategy π_2 cannot be improved further, since $\pi_1(2) = 3 > 2 = \mu\pi_2$. Unfortunately, 2 is not the least solution. The problem here stems from the fact that the function π_1 is not non-expansive in the sup-norm, i.e., it does not hold $\|f(x) - f(y)\|_\infty \leq \|x - y\|_\infty$ for all $x, y \in \mathbb{R}$ (see Adjé et al. (2008) for more details). □

Although minimality of the obtained solution cannot be guaranteed in general, there are indeed important cases where minimality can be guaranteed by an enhanced $\wedge$-strategy improvement step. Adjé et al. (2008) describe how to guarantee minimality for the case that all mappings are non-expansive. A notable advantage of the $\wedge$-strategy iteration approach is that it can be stopped at anytime with a safe over-approximation. It thus can give us non-trivial safe results, even if the set of $\wedge$-strategies is infinite.

5.2. Least fixpoints for max-affine self-maps

In this subsection, we explain how to compute the least solution of a system $\mathcal{C}$ of inequalities of the form $\mathbf{x}_i \geq f(\mathbf{x}_1, \ldots, \mathbf{x}_n)$, where $\mathbf{x}_1, \ldots, \mathbf{x}_n$ are distinct variables and $f : \overline{\mathbb{R}}^n \to \overline{\mathbb{R}}$ is monotone and affine. A function $f : \overline{\mathbb{R}}^n \to \overline{\mathbb{R}}$ is monotone and affine iff there exist $c_0 \in \overline{\mathbb{R}}$ and $c_1, \ldots, c_n \in \mathbb{R}_{\geq 0}$ such that $f(x_1, \ldots, x_n) = c_0 + c_1 \cdot x_1 + \cdots + c_n \cdot x_n$ for all $x_1, \ldots, x_n \in \overline{\mathbb{R}}$. Here, we use the convention $-\infty + \infty = -\infty$.

For simplicity we assume that the least solution ρ^* of $\mathcal{C}$ maps every variable to a value that is strictly greater than $-\infty$, i.e., $\rho^*(\mathbf{x}) > -\infty$ for all variable $\mathbf{x}$. We can do so w.l.o.g. for the following reason: We can determine the variables that are $-\infty$ in the least solution by performing n Kleene iteration steps (recall that $n = |\mathbf{X}|$ is the number of variables). That is, for all variables $\mathbf{x}$, $\rho^*(\mathbf{x}) > -\infty$ iff $\rho^{(n)}(\mathbf{x}) > -\infty$, where $\rho^{(n)}$ denotes the n-th Kleene approximate. This can be shown by considering the fixpoint iterations, using upward-chain-continuity of the right-hand sides and the fact that, for all $c_1, \ldots, c_n \in \mathbb{R}_{>0}$, $c_1 \cdot x_1 + \cdots + c_k \cdot x_k > -\infty$ iff $x_i > -\infty$ for all $i \in \{1, \ldots, k\}$. Finally, we can remove the variables $\mathbf{x}$ with $\rho^*(\mathbf{x})$ and the corresponding inequalities from the system of inequalities.

In order to deal with variables that are ∞, we process one strongly connected component after the other. Let $\mathbf{X}$ denote the set of variables, and $\mathcal{G} = (\mathbf{X}, \to)$ be the *variable dependency graph* of the system $\mathcal{C}$ of inequalities, i.e., the nodes of $\mathcal{G}$ are the variables of $\mathcal{E}$ and we write $\mathbf{x}_i \to \mathbf{x}_j$ iff $\mathbf{x}_i = \infty$ implies $\mathbf{x}_j = \infty$, i.e., if there exists an inequality $\mathbf{x}_j \geq f(\mathbf{x}_1, \ldots, \mathbf{x}_n)$ with $f(x_1, \ldots, x_n) = c_0 + c_1 \cdot x_1 + \cdots + c_n \cdot x_n$, where $c_0 > -\infty$ and $c_i > 0$. If $\mathcal{G}$ is strongly connected, then the least solution ρ^* of $\mathcal{C}$ can be determined by solving the following linear programming problem:

$$\min \sum_{i=1}^{n} \mathbf{x}_i \text{ subject to} \tag{55}$$

$$\mathbf{x}_i \geq f(\mathbf{x}_1, \ldots, \mathbf{x}_n) \quad \text{for all inequalities } \mathbf{x}_i \geq f(\mathbf{x}_1, \ldots, \mathbf{x}_n) \text{ of } \mathcal{C}. \tag{56}$$

The above linear program aims at minimizing the sum of all variables $\mathbf{x} \in \mathbf{X}$. The feasible space is simply the set of all solutions of the system $\mathcal{C}$ of inequalities. If this linear program is infeasible, then $\rho^*(\mathbf{x}) = \infty$ holds for all variables $\mathbf{x} \in \mathbf{X}$. If this linear program is feasible, then ρ^* is the uniquely determined optimal solution. That is, If the variable dependency graph of $\mathcal{C}$ is strongly connected, then the least solution of $\mathcal{C}$ can be computed by solving a linear programming problem that can be constructed in linear time.

For computing the least solution in case that the variable dependency graph $\mathcal{G}$ of $\mathcal{C}$ is not strongly connected, we divide the system of inequalities into strongly connected components. The goal of dividing the system $\mathcal{C}$ into strongly connected components is to find finite solutions to subsystems even if the complete system does not contain a finite solution.

We start with an arbitrary non-trivial strongly connected component without incoming edges. According to the above observations, the least solution of the induced system of inequalities can be computed by solving a linear programming problem that can be constructed in linear time. After we have determined the values for this strongly connected component, we can replace these variables with their values. The above procedure is repeated until all strongly connected components are solved. Since the number of strongly connected components is bounded by the number of variables, we get:

Theorem 19 (*Gaubert et al., 2007*)**.** *The least solution of a system of inequalities of the form* $\mathbf{x}_i \geq f(\mathbf{x}_1, \ldots, \mathbf{x}_n)$*, where* f *is a monotone and affine operator, can be computed by solving linearly many linear programming problems, each of which can be constructed in linear time. Thus, it can be computed in polynomial time.* □

Example 20. We consider the following system of inequalities:

$$\mathbf{x}_1 \geq -10 \qquad \mathbf{x}_1 \geq \frac{1}{4} \cdot \mathbf{x}_2 + 1 \qquad \mathbf{x}_2 \geq 2 \cdot \mathbf{x}_1 \qquad \mathbf{x}_3 \geq \mathbf{x}_3 + \mathbf{x}_1 - 1 \qquad \mathbf{x}_3 \geq 0. \tag{57}$$

Our goal is to compute the least solution using the method presented in this subsection. The strongly connected components of the variable dependency graph are $\{\mathbf{x}_1, \mathbf{x}_2\}$ and $\{\mathbf{x}_3\}$. The variables $\mathbf{x}_1$ and $\mathbf{x}_2$ do not depend on the variable $\mathbf{x}_3$. Thus, in the first step we have to compute the uniquely determined optimal solution of the following linear programming problem:

$$\min\ \mathbf{x}_1 + \mathbf{x}_2 \qquad \mathbf{x}_1 \geq -10 \qquad \mathbf{x}_1 \geq \frac{1}{4} \cdot \mathbf{x}_2 + 1 \qquad \mathbf{x}_2 \geq 2 \cdot \mathbf{x}_1. \tag{58}$$

The uniquely determined optimal solution gives us $\mathbf{x}_1 = 2$ and $\mathbf{x}_2 = 4$. After substituting the variables $\mathbf{x}_1$ and $\mathbf{x}_2$ with their values, it remains to compute the least solution of the following system of inequalities:

$$\mathbf{x}_3 \geq \mathbf{x}_3 + 2 - 1 \vee 0 \tag{59}$$

Thus, we have to determine the uniquely determined optimal solution of the following linear programming problem:

$$\min\ \mathbf{x}_3 \qquad \mathbf{x}_3 \geq \mathbf{x}_3 + 1 \qquad \mathbf{x}_3 \geq 0. \tag{60}$$

This linear programming problem is infeasible. Thus, we get $\mathbf{x}_3 = \infty$. Hence, the least solution of the original system of inequalities is $\mathbf{x}_1 = 2, \mathbf{x}_2 = 4$, and $\mathbf{x}_3 = \infty$. □

When we use interior point methods for solving the linear programming problems, we obtain a polynomial-time algorithm. However, the number of arithmetic operations and memory accesses then depends on the sizes of the occurring numbers. Thus, the algorithm is not uniform. A uniform polynomial-time algorithm is not known.

5.3. $\wedge$-strategies for systems of concave equations

We now aim at specializing our $\wedge$-strategy improvement algorithm to an algorithm for solving systems of $\vee$-cmcave equations as introduced in Section 4. For that, let us consider the following system of $\vee$-cmcave equations:

$$\begin{aligned} \mathbf{x}_1 &= f_{1,1}(\mathbf{x}_1, \ldots, \mathbf{x}_n) \vee \cdots \vee f_{1,k_1}(\mathbf{x}_1, \ldots, \mathbf{x}_n) \\ &\vdots \\ \mathbf{x}_n &= f_{n,1}(\mathbf{x}_1, \ldots, \mathbf{x}_n) \vee \cdots \vee f_{n,k_n}(\mathbf{x}_1, \ldots, \mathbf{x}_n). \end{aligned} \tag{61}$$

In a first step, we construct a set Π of $\wedge$-strategies for the function

$$f = \begin{pmatrix} f_{1,1} \vee \cdots \vee f_{1,k_1} \\ \vdots \\ f_{n,1} \vee \cdots \vee f_{n,k_n} \end{pmatrix} : \overline{\mathbb{R}}^n \to \overline{\mathbb{R}}^n. \tag{62}$$

We want to construct Π in such a way that the least fixpoint $\mu\pi$ of π can be computed efficiently for every $\wedge$-strategy $\pi \in \Pi$. Firstly, we define the set $\mathcal{T}_{k,1}$ as the set of all monotone and affine functions $f : \overline{\mathbb{R}}^k \to \overline{\mathbb{R}}$, i.e. functions of the form $f(x) = a^\top x + b$ with $a \in \mathbb{R}^k_{\geq 0}$ and $b \in \overline{\mathbb{R}}$. The set $\mathcal{T}^\vee_{k,1}$ is then defined by $\mathcal{T}^\vee_{k,1} := \{f_1 \vee \cdots \vee f_l \mid f_1, \ldots, f_l \in \mathcal{T}_{k,1}\}$. The set $\mathcal{T}^\vee_{k,m}$ is finally defined by $\mathcal{T}^\vee_{k,m} := \{(f_1, \ldots, f_m)^\top \mid f_1, \ldots, f_m \in \mathcal{T}^\vee_{k,1}\}$. For all operators $f : \overline{\mathbb{R}}^k \to \overline{\mathbb{R}}^m$ and all $x \in \overline{\mathbb{R}}^k$, we define the operator $f \nearrow^\infty_x : \overline{\mathbb{R}}^k \to \overline{\mathbb{R}}^m$ by

$$f \nearrow^\infty_x (y) = \begin{cases} f(y) & \text{if } y \leq x \\ (\infty, \ldots, \infty)^\top & \text{if } y > x \end{cases} \qquad \text{for all } y \in \overline{\mathbb{R}}^k. \tag{63}$$

Let now $f = (f_1, \ldots, f_n)^\top$, where $f_1, \ldots, f_n : \overline{\mathbb{R}}^n \to \overline{\mathbb{R}}$ are monotone and concave. We define the set Π of $\wedge$-strategies for f as follows:

$$\Pi = \{\pi \mid \exists \pi' : \mathcal{T}^{\vee}_{n,n} . \exists x \in \overline{\mathbb{R}}^n . \pi = \pi' \nearrow^{\infty}_{x} \text{ and } \pi \geq f\}. \tag{64}$$

Each $\wedge$-strategy $\pi \in \Pi$ is of the form $\pi' \nearrow^{\infty}_{x}$, where π' is a maximum of finitely many monotone and affine self-maps. The $\wedge$-strategy π is thus in particular convex. As we have seen in Section 5.2, the least fixpoint $\mu\pi'$ of π' and thus the least fixpoint $\mu\pi$ of π can be computed through linear programming. Because $f \leq \pi$ for all $\pi \in \Pi$, it follows that $\mu f \leq \mu\pi$ for all $\pi \in \Pi$. Moreover, because each component of f is a maximum of finitely many monotone functions, it follows that for each $x \in \overline{\mathbb{R}}^n$, there exists some $\pi^* \in \Pi$ with $\pi^*(x) = f(x)$. The most trivial choice for π^* is $\pi^* = \pi' \nearrow^{\infty}_{x}$ with $\pi'(y) = f(x)$ for all $y \in \overline{\mathbb{R}}^n$. We thus get $\mu f = \min\{\mu\pi \mid \pi \in \Pi\}$ as desired.

Note that Π can be indeed infinite. Nonetheless, in some cases there exists a finite subset Π' of Π such that $f(x) = \min\{\pi(x) \mid \pi \in \Pi'\}$ for all $x \in \overline{\mathbb{R}}^n$. One important example for this is the case that all $f_{i,j}$'s are not arbitrary monotone and concave functions, but just minima of finitely many monotone and affine functions (see Gaubert et al., 2007). Then we can restrict our considerations to a finite subset.

5.4. Improving $\wedge$-strategies

We now explain how the $\wedge$-strategy improvement step (Step 4) can be realized. We assume that we are given a post-fixpoint x of f, i.e., $x \geq f(x)$. Our goal is to compute a "non-trivial" $\wedge$-strategy $\pi \in \Pi$ such that $\pi(x) = f(x)$. As discussed in Section 5.3, the most trivial choice for π is $\pi = \pi' \nearrow^{\infty}_{x}$ with $\pi'(y) = f(x)$ for all $y \in \overline{\mathbb{R}}^n$. Then, however, the $\wedge$-strategy algorithm would perform exactly a Kleene iteration. In order to increase the speed of convergence, it is important to determine a "good" $\wedge$-strategy π.

For a start, let us assume that, for any monotone and concave function $f : \overline{\mathbb{R}}^k \to \overline{\mathbb{R}}$ and any $x \in \overline{\mathbb{R}}^k$, we can compute a "good" monotone and affine function $T_{f,x} : \overline{\mathbb{R}}^k \to \overline{\mathbb{R}}$ such that $T_{f,x}(y) \geq f(y)$ for all $y \leq x$, and $T_{f,x}(x) = f(x)$. Such a function $T_{f,x}$ exists, because f is monotone.

Then, for $f = f_1 \vee \cdots \vee f_k$ with $f_1, \ldots, f_k : \overline{\mathbb{R}}^k \to \overline{\mathbb{R}}$ monotone and concave and $x \in \overline{\mathbb{R}}^k$, we define the monotone function $T_{f,x} : \overline{\mathbb{R}}^k \to \overline{\mathbb{R}}$ by

$$T_{f,x}(y) := T_{f_1,x}(y) \vee \cdots \vee T_{f_k,x}(y) \quad \text{for all } y \in \overline{\mathbb{R}}^k. \tag{65}$$

For $f = (f_1, \ldots, f_n)^T$ with monotone $f_1, \ldots, f_n : \overline{\mathbb{R}}^k \to \overline{\mathbb{R}}$, we finally define the monotone function $T_{f,x} : \overline{\mathbb{R}}^k \to \overline{\mathbb{R}}^n$ by

$$T_{f,x}(y) = (T_{f_1,x}(y), \ldots, T_{f_n,x}(y))^\top \quad \text{for all } y \in \overline{\mathbb{R}}^k. \tag{66}$$

In consequence, $T_{f,x}(y) \geq f(y)$ for all $y \in \overline{\mathbb{R}}^k$ and $T_{f,x}(x) = f(x)$. Therefore, for a monotone and concave self-map $f : \overline{\mathbb{R}}^n \to \overline{\mathbb{R}}^n$ and some $x \in \overline{\mathbb{R}}^n$, $\pi := T_{f,x} \nearrow^{\infty}_{x}$ is a $\wedge$-strategy for f with $\pi(x) = f(x)$.

It remains to answer the question how a "good" monotone and affine function $T_{f,x} : \overline{\mathbb{R}}^n \to \overline{\mathbb{R}}$ can be computed for some given monotone and concave function $f : \overline{\mathbb{R}}^n \to \overline{\mathbb{R}}$ and some given $x \in \overline{\mathbb{R}}^n$ with $-\infty < f(x) < \infty$ such that $T_{f,x}(y) \geq f(y)$ for all $y \leq x$ and $T_{f,x}(x) = f(x)$. In order to obtain a reasonable quality, we search for monotone and affine function $T_{f,x}$ that fulfills a stronger requirement. We want $T_{f,x}(y) \geq f(y)$ to hold not only for all $y \leq x$, but for all $y \in \overline{\mathbb{R}}^n$. However, the existence of a monotone and affine $T_{f,x}$ with $T_{f,x}(y) \geq f(y)$ for all $y \in \overline{\mathbb{R}}^n$ and $T_{f,x}(x) = f(x)$ is not guaranteed. An example is given by the monotone and concave function $f = \sqrt{\cdot}$ at $x = 0$. These degenerate cases can, however, be detected. In these cases, we still can always choose a trivial monotone and affine $T_{f,x}$.

Let us now assume that $x \in \mathbb{R}^n$. We can do so w.l.o.g., since all other components can be removed. We aim at computing a monotone and affine function $T_{f,x} : \overline{\mathbb{R}}^n \to \overline{\mathbb{R}}$ such that $T_{f,x}(x) = f(x)$ and

$T_{f,x}(y) \geq f(y)$ for all $y \in \overline{\mathbb{R}}^n$. Hence,

$$T_{f,x}(y) = f(x) + d^\top (y - x) \quad \text{for all } y \in \mathbb{R}^n \tag{67}$$

for some $d \in \mathbb{R}^n_{\geq 0}$, and

$$f(y) - d^\top (y - x) \leq f(x) \quad \text{for all } y \in \mathbb{R}^n. \tag{68}$$

Such a $d \in \mathbb{R}^n_{\geq 0}$ can be determined by means of convex optimization. Let

$$g(d, y) := f(y) - d^\top (y - x), \quad \text{and} \quad g(d) := \sup \{g(d, y) \mid y \in \mathbb{R}^n\}$$

for all $d \in \mathbb{R}^n_{\geq 0}$ and all $y \in \mathbb{R}^n$. For all $d \in \mathbb{R}_{\geq 0}$, condition (68) is fulfilled iff

$$g(d) \leq f(x). \tag{69}$$

Therefore, we search for a $d \in \mathbb{R}^n_{\geq 0}$ that minimizes the function g. We can do this through convex optimization, if g is convex and we have a method for evaluating g. Indeed, the function g is convex, since it is the point-wise supremum of a set of affine functions. For computing the value $g(d)$ for a given $d \in \mathbb{R}^n_{\geq 0}$, we have to solve an unconstrained convex optimization problem, because $g(d, \cdot)$ is concave and thus $-g(d, \cdot)$ is convex.

There are cases, where a $d \in \mathbb{R}^n_{\geq 0}$ that minimizes the function g does not exist. An example is given by the monotone and concave function $f = \sqrt{\cdot}$ and $x = 0$. For this example $g(d)$ converges to 0 if d converges to ∞, but there does not exists some $d \in \mathbb{R}_{\geq 0}$ such that $g(d) = 0$. However, this is rather a theoretical problem than a problem of practical relevance. In practice, one simply chooses some $d \in \mathbb{R}^n_{\geq 0}$ such that $g(d)$ is close to the optimal value $\inf\{g(d') \mid d' \in \mathbb{R}_{\geq 0}\}$. Even if we chose $d = (0, \ldots, 0)^\top$, which would be the trivial choice, we would at least guarantee the progress that is also obtained by a Kleene iteration step.

Within our application described in Section 3, the monotone and affine function $T_{f,x}$ can be computed more efficiently. There, the functions $f : \overline{\mathbb{R}}^n \to \overline{\mathbb{R}}$ are of the form

$$f(x) = \sup \{C \bullet X \mid X \succeq 0, \mathcal{A}(X) = a, \mathcal{B}(X) \leq x\}, \tag{70}$$

i.e., $f(x)$ is given by the optimal value of the SDP problem:

$$\max_X \; C \bullet X \quad \mathcal{A}(X) = a \quad \mathcal{B}(X) \leq x \quad X \succeq 0. \tag{71}$$

For simplicity, let us assume that $x \in \mathbb{R}^n$. The other cases can be dealt with by removing constraints from the semi-definite programming problem. We use the *dual* problem to compute $T_{f,x}$. The dual problem (see e.g. Todd, 2001) is given by:

$$\min_{\lambda,\mu} \quad x^\top \lambda + a^\top \mu \quad \mathcal{B}\lambda + \mathcal{A}\mu \succeq C \quad \lambda \geq 0. \tag{72}$$

Let $d(x)$ denote the optimal value of the dual problem. Weak duality gives us $f(x) \leq d(x)$. In particular, we thus get $-\infty < d(x)$. We define the monotone and affine function $T_{f,x}$ as follows: If $d(x) = \infty$, i.e., if the dual is infeasible, then we set

$$T_{f,x}(z) = f(x) \quad \text{for all } z \in \overline{\mathbb{R}}^n. \tag{73}$$

If the dual has an optimal solution (λ, μ), then we define the hyperplane $T_{f,x}$ by

$$T_{f,x}(z) = \lambda^\top z + \mu^\top a \quad \text{for all } z \in \overline{\mathbb{R}}^n. \tag{74}$$

If the dual is feasible, but has no optimal solution, then we choose *any* good feasible solution. Then weak duality guarantees that $T_{f,x} \geq f$. The affine function $T_{f,x}$ is monotone, because $\lambda \geq 0$.

In order to conclude $T_{f,x}(x) = f(x)$, we require stronger assumptions, for instance, assumptions that imply strong duality. One sufficient criterion for strong duality and the existence of an optimal solution for the dual problem is that all components of $\mathcal{A}$ are linearly independent and $\{X \succ 0 \mid \mathcal{A}(X) = a,\ \mathcal{B}(X) \lhd x\} \neq \emptyset$ (cf. Todd, 2001).

The result of the above discussion can be summarized as follows: A monotone and affine self-map $T_{f,x}$ such that $T_{f,x}(y) \geq f(y)$ for all $y \in \overline{\mathbb{R}}^n$ and $T_{f,x}(x) = f(x)$ can be computed through semi-definite programming, whenever the above sufficient condition for strong duality is fulfilled. Again, this is rather a theoretical problem than a problem of practical relevance. In the case that $T_{f,x}(x) = f(x)$ is not fulfilled (i.e., we have $T_{f,x}(x) > f(x)$), when using the $T_{f,x}$ we have obtained through semi-definite programming, we can simply redefine $T_{f,x}$ by $T_{f,x}(z) = f(x)$ for all $z \in \overline{\mathbb{R}}^n$.

We can summarize the results obtained so far in the following theorem:

Theorem 21. *The $\wedge$-strategy improvement algorithm (Algorithm 1) can be applied for solving systems of $\vee$-cmcave equations. The algorithm starts with a given post-solution and constructs a decreasing sequence of post-solutions. Each $\wedge$-strategy improvement step (Step 4 in Algorithm 1) can be performed through convex optimization, where one solves a convex optimization problem for each right-hand side. In the application described in Section 3, the right-hand sides are of the form (70). In consequence, each $\wedge$-strategy improvement step can be performed more efficiently through semi-definite programming. Each value determination step (Step 4 in Algorithm 1) can be performed through linear programming (cf. Section 5.2). The $\wedge$-strategy improvement algorithm can be stopped at any time with a safe over-approximation.* □

5.5. The harmonic oscillator

We continue with Example 12. In order to analyze the harmonic oscillator, we solve the following systems of equations:

$$\mathbf{x}_{\mathbf{st},p_1} = 0 \vee ([\![s]\!]^{\mathcal{R}}\{p \mapsto \mathbf{x}_{\mathbf{st},p} \mid p \in P\})(p_1) \tag{75}$$

$$\mathbf{x}_{\mathbf{st},p_2} = 1 \vee ([\![s]\!]^{\mathcal{R}}\{p \mapsto \mathbf{x}_{\mathbf{st},p} \mid p \in P\})(p_2) \tag{76}$$

$$\mathbf{x}_{\mathbf{st},p_3} = 0 \vee ([\![s]\!]^{\mathcal{R}}\{p \mapsto \mathbf{x}_{\mathbf{st},p} \mid p \in P\})(p_3) \tag{77}$$

$$\mathbf{x}_{\mathbf{st},p_4} = 1 \vee ([\![s]\!]^{\mathcal{R}}\{p \mapsto \mathbf{x}_{\mathbf{st},p} \mid p \in P\})(p_4) \tag{78}$$

$$\mathbf{x}_{\mathbf{st},p_5} = 7 \vee ([\![s]\!]^{\mathcal{R}}\{p \mapsto \mathbf{x}_{\mathbf{st},p} \mid p \in P\})(p_5). \tag{79}$$

We emphasize that the right-hand sides are $\vee$-cmcave expressions. It is easy to verify that $\mathbf{x}_{\mathbf{st},p_1} = \cdots = \mathbf{x}_{\mathbf{st},p_4} = \infty$, $\mathbf{x}_{\mathbf{st},p_5} = 7$ is a post-solution. In order to simplify notations, let $c_1 = c_3 = 0$, $c_2 = c_4 = 1$, $c_5 = 7$, and, for all $i \in \{1, \ldots, 5\}$, $f_i : \overline{\mathbb{R}}^n \to \overline{\mathbb{R}}$ be defined by

$$f_i((x_1, \ldots, x_5)^\top) := c_i \vee ([\![s]\!]^{\mathcal{R}}\{p_i \mapsto x_i \mid i \in \{1, \ldots, 5\}\})(p_i). \tag{80}$$

Let moreover $f = (f_1, \ldots, f_5)^\top$, i.e., f denotes the right-hand side of the above system of equations. If we evaluate the right-hand sides, we get

$$f((\infty, \infty, \infty, \infty, 7)^\top) \simeq (2.0426, 2.0426, 1.6651, 1.6651, 7)^\top. \tag{81}$$

For evaluating the right-hand sides, we can use semi-definite programming. Many state of the art implementations (e.g. CSDP Borchers, 1999a,b; Borchers and Young, 2007, SeDuMi (Sturm, 1999), SDPA Yamashita et al., 2003a,b; Fukuda et al., 2007, and SDPT3 Tohet al., 1999; Tütüncü et al., 2003) are based on primal–dual interior point methods (see e.g. Wright (1997) for more information on primal–dual interior point methods). They solve the primal and the dual problem at the same time. From a dual optimal solution, we obtain the first $\wedge$-strategy $\pi^{(0)}$ that is given as follows:

$$\pi^{(0)} := T_{f,(\infty,\infty,\infty,\infty,7)^\top}((x_1, \ldots, x_5)^\top) \simeq \begin{pmatrix} 0 \vee 0.14588 \cdot x_5 + 1.0214 \\ 1 \vee 0.14588 \cdot x_5 + 1.0214 \\ 0 \vee 0.11892 \cdot x_5 + 0.83263 \\ 1 \vee 0.11892 \cdot x_5 + 0.83263 \\ 7 \vee 0.99456 \cdot x_5 \end{pmatrix} \tag{82}$$

We now explain how we have obtained the first component. According to the findings from Example 12, we have

$$(\llbracket s \rrbracket^{\mathcal{R}}\{p_1 \mapsto \infty, p_2 \mapsto \infty, p_3 \mapsto \infty, p_4 \mapsto \infty, p_5 \mapsto 7\})(p_1) \tag{83}$$

$$= \sup\{C_1 \bullet X \mid X \succeq 0, X_{1\cdot 1} = 1, B_5 \bullet X \leq 7\} \tag{84}$$

$$= \sup\left\{ \begin{pmatrix} 0 & -0.5 & -0.005 \\ -0.5 & 0 & 0 \\ -0.005 & 0 & 0 \end{pmatrix} \bullet X \mid X \succeq 0, X_{1\cdot 1} = 1, \begin{pmatrix} 0 & 0 & 0 \\ 0 & 2 & 1 \\ 0 & 1 & 3 \end{pmatrix} \bullet X \leq 7 \right\} \tag{85}$$

$$= 2.0426. \tag{86}$$

We have seen in Section 5.4, that, in order to compute an affine over-approximation of $(\llbracket s \rrbracket^{\mathcal{R}}\{p_i \mapsto x_i \mid i \in \{1, \ldots, 5\}\})(p_i)$ that is exact at $x_1 = x_2 = x_3 = x_4 = \infty$ and $x_5 = 7$, we can solve the dual problem that is given as follows:

$$\inf\left\{7\lambda + \mu \mid \lambda \geq 0,\ \mu \in \mathbb{R},\ \lambda B_5 + \mu \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \succeq C_1\right\} \tag{87}$$

$$= \inf\{7\lambda + \mu \mid \lambda \geq 0,\ \mu \in \mathbb{R}, \tag{88}$$

$$\lambda \begin{pmatrix} 0 & 0 & 0 \\ 0 & 2 & 1 \\ 0 & 1 & 3 \end{pmatrix} + \mu \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \succeq \begin{pmatrix} 0 & -0.5 & -0.005 \\ -0.5 & 0 & 0 \\ -0.005 & 0 & 0 \end{pmatrix}\Bigg\}. \tag{89}$$

Running a semi-definite programming solver, e.g. CSDP or SeDuMi, gives us the result $\lambda \simeq 0.14588$ and $\mu \simeq 1.0214$. This gives us the first component of $\pi^{(0)}$. The remaining components can be computed in the same way.

As described in Section 5.2 the least fixpoint of $\pi^{(0)}$ can be computed through linear programming. We get

$$x^{(0)} := \mu\pi^{(0)} = (2.0426, 2.0426, 1.6651, 1.6651, 7)^{\top}. \tag{90}$$

Then, by again solving semi-definite and linear programming problems, we get

$$\pi^{(1)} := T_{f,x^{(0)}}((x_1, \ldots, x_5)^{\top}) \simeq \begin{pmatrix} 0 \vee 0.90541 \cdot x_1 + 0.01340 \cdot x_5 + 0.093820 \\ 1 \vee 0.90541 \cdot x_2 + 0.01340 \cdot x_5 + 0.093819 \\ 0 \vee 0.88297 \cdot x_3 + 0.01346 \cdot x_5 + 0.094205 \\ 1 \vee 0.88297 \cdot x_4 + 0.01346 \cdot x_5 + 0.094205 \\ 7 \vee 0.99456 \cdot x_5 \end{pmatrix} \tag{91}$$

and

$$x^{(1)} := \mu\pi^{(1)} \simeq (1.9838, 1.9838, 1.6098, 1.6098, 7.0000)^{\top}. \tag{92}$$

Continuing this process, we find:

$$x^{(2)} := \mu\pi^{(2)} \simeq (1.8971, 1.8971, 1.5434, 1.5434, 7.0000)^{\top} \tag{93}$$

$$x^{(3)} := \mu\pi^{(3)} \simeq (1.8718, 1.8718, 1.5280, 1.5280, 7.0000)^{\top} \tag{94}$$

$$x^{(4)} := \mu\pi^{(4)} \simeq (1.8708, 1.8708, 1.5275, 1.5275, 7.0000)^{\top} \tag{95}$$

$$x^{(5)} := \mu\pi^{(5)} \simeq (1.8708, 1.8708, 1.5275, 1.5275, 7.0000)^{\top}. \tag{96}$$

The $\wedge$-strategy iteration stabilizes after a few iterations. The run of the $\wedge$-strategy improvement algorithm is visualized in Fig. 6. As a result, we obtain

$$\mu f \leq (1.8708, 1.8708, 1.5275, 1.5275, 7.0000)^{\top}. \tag{97}$$

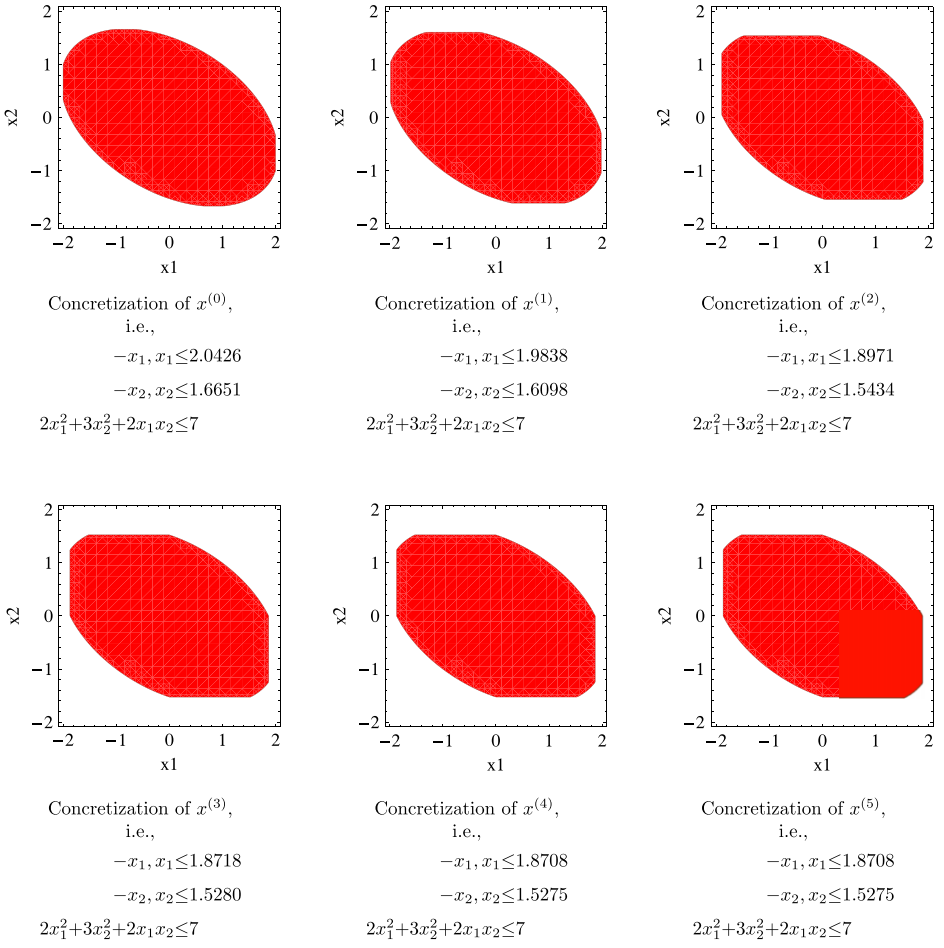


Fig. 6. Visualization of a run of the $\wedge$-strategy iteration algorithm for the harmonic oscillator from Section 2.

Therefore, the following invariants hold at program point **st**:

$$-x_1 \leq 1.8708 \qquad x_1 \leq 1.8708 \qquad -x_2 \leq 1.5275 \tag{98}$$

$$x_2 \leq 1.5275 \qquad 2x_1^2 + 3x_2^2 + 2x_1x_2 \leq 7. \tag{99}$$

6. The max-strategy iteration approach

Before giving a formal description of the max-strategy iteration approach in Section 6.2, we explain it by a simple example in Section 6.1. In Section 6.3 we apply the max-strategy iteration approach to the harmonic oscillator as introduced in Section 2.

6.1. A simple example

Our goal is to compute the least solution of the following equation system:

$$\mathbf{x} = 0.4 \vee \sqrt{\mathbf{x}} \vee 1 + \sqrt{\mathbf{x} - 1}. \tag{100}$$

Here, $\sqrt{x} = \sup\{y \in \mathbb{R} \mid y^2 \leq x\}$ for all $x \in \overline{\mathbb{R}}$. Note that $\sqrt{x} = \sup \emptyset = -\infty$ for all $x < 0$. The important property is that all right-hand sides are $\vee$-cmcave. The graph of the left-hand side **x** and

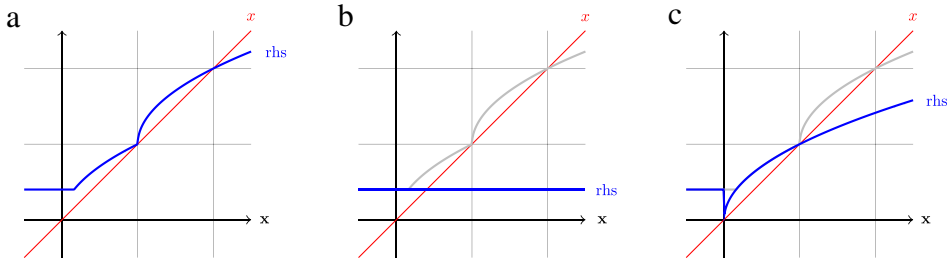


Fig. 7. A run of the $\vee$-strategy improvement algorithm.

the graph of the right-hand side $0.4 \vee \sqrt{\mathbf{x}} \vee 1 + \sqrt{\mathbf{x} - 1}$ are drawn in Fig. 7.(a). The least solution is the least x-coordinate where the two graphs cross, i.e., it is given by 1.

We now use the $\vee$-strategy improvement algorithm of Gawlitza and Seidl (2007a,b, in press, 2010) for finding the least solution. We consider the computation of the least solution as a competition between a maximizer and a minimizer. The current approximate to the least solution is the current state of the play. The maximizer aims at maximizing the current approximate whereas the minimizer aims at minimizing it. In some state, i.e., at some approximate the maximizer is allowed to select an argument of the finite maximum $0.4 \vee \sqrt{\mathbf{x}} \vee 1 + \sqrt{\mathbf{x} - 1}$, for instance $\sqrt{\mathbf{x}}$. Such a selection is called a $\vee$-*strategy*. The play starts at the approximate $-\infty$. This is the current state of the play at the beginning. At this point, the most profitable $\vee$-strategy is the argument 0.4, since $\sqrt{\mathbf{x}}$ and $1 + \sqrt{\mathbf{x} - 1}$ evaluate to $-\infty$. The play proceeds by performing a least fixpoint iteration starting at the current state $-\infty$ using the current $\vee$-strategy, i.e., the next approximate is the least solution of the equation system

$$\mathbf{x} = 0.4 \tag{101}$$

that exceeds $-\infty$. Hence, the next approximate is 0.4 (cf. Fig. 7.(b)). Note that 0.4 is not only the least solution of the equation system $\mathbf{x} = 0.4$ that exceeds 0.4, but it is also the greatest solution of the inequation $\mathbf{x} \leq 0.4$, i.e., the greatest point in the convex area that is above the graph of the left-hand side and below the graph of the concave right-hand side (cf. Fig. 7.(b)). This is not by accident. During a run of our $\vee$-strategy improvement algorithm, the next approximate is always the greatest point of such a convex area. In consequence, it can be computed through algorithms for solving convex optimization problems.

Now, we try to improve the current $\vee$-strategy locally at 0.4. Since $\sqrt{0.4} > 0.4$ holds, we can improve the current $\vee$-strategy to the $\vee$-strategy that selects the argument $\sqrt{\mathbf{x}}$.[3] This gives us a *strict local improvement*. Thus, the next approximate is the least solution of the equation system

$$\mathbf{x} = \sqrt{\mathbf{x}} \tag{102}$$

that exceeds 0.4. Hence, the next approximate is 1 (cf. Fig. 7.(c)). It is again the greatest solution of the inequation system $\mathbf{x} \leq \sqrt{\mathbf{x}}$. Thus, it is the uniquely determined optimal solution of the following convex optimization problem:

$$\max \ \mathbf{x} \quad \text{subject to } \mathbf{x}^2 - \mathbf{x} \leq 0. \tag{103}$$

In this case the unique optimal solution can for instance be computed through semi-definite programming, because it is a *convex quadratic optimization problem*.

Accordingly, our current approximate is 1 and our current $\vee$-strategy selects the argument $\sqrt{\mathbf{x}}$. We again try to improve the current $\vee$-strategy, i.e., we search for a $\vee$-strategy that is strictly more profitable at the current approximate 1 than the current $\vee$-strategy. Since $0.4 < 1 = 1 + \sqrt{1 - 1} = 1 = \sqrt{1}$ holds, there is no such $\vee$-strategy. In other words, the current $\vee$-strategy cannot be improved at the current approximate (cf. Fig. 7.(c)). This means that we have found a solution of the

[3] Since $1 + \sqrt{0.4 - 1} = 1 + \sqrt{-0.6} = 1 + -\infty = -\infty$ holds, a switch to the $\vee$-strategy that selects the argument $1 + \sqrt{\mathbf{x} - 1}$ is not profitable at the approximate 0.4.

equation system

$$\mathbf{x} = 0.4 \vee \sqrt{\mathbf{x}} \vee 1 + \sqrt{\mathbf{x} - 1}. \tag{104}$$

Since the sequence of approximates is monotonically increasing and bounded by the least solution, the least solution has been found. In summary: The $\vee$-strategy improvement algorithm terminates and returns the least solution 1.

6.2. The max-strategy improvement algorithm

In this section we compute least solutions of systems of $\vee$-cmcave equations through the $\vee$-strategy improvement algorithm of Gawlitza and Seidl (2007a,b, 2008, in press). Systems of $\vee$-cmcave equations are in particular systems of monotone equations over the *complete linearly ordered set* $\overline{\mathbb{R}}$. For the sake of generality, we subsequently consider an arbitrary complete linearly ordered set.

A *$\vee$-strategy* σ for a system $\mathcal{E}$ of monotone equations over a complete linearly ordered set is a function that maps every expression $e_1 \vee \cdots \vee e_k$ occurring in $\mathcal{E}$ to one of the immediate sub-expressions $e_j, j \in \{1, \ldots, k\}$. We denote the set of all $\vee$-strategies for $\mathcal{E}$ by $\Sigma_{\mathcal{E}}$. We drop the subscript, whenever it is clear from the context. Finally, we set $\mathcal{E}(\sigma) := \{\mathbf{x} = \sigma(e) \mid \mathbf{x} = e \in \mathcal{E}\}$.

Example 22. For the system $\mathcal{E} = \{\mathbf{x} = \frac{1}{2} \vee \sqrt{\mathbf{x}}\}$ of $\vee$-cmcave equations and the $\vee$-strategy $\sigma = \{\frac{1}{2} \vee \sqrt{\mathbf{x}} \mapsto \frac{1}{2}\}$, we have $\mathcal{E}(\sigma) = \{\mathbf{x} = \frac{1}{2}\}$. □

The $\vee$-strategy improvement algorithm iterates over $\vee$-strategies. It maintains a current $\vee$-strategy and a current *approximate* to the least solution. In each step, if possible, the current $\vee$-strategy is improved w.r.t. the current approximate, and a new current approximate is computed w.r.t. the new current $\vee$-strategy and the current approximate:

Definition 23 (*Improvements*)**.** Let $\mathcal{E}$ be a system of monotone equations over a complete linearly ordered set. Let $\sigma, \sigma' \in \Sigma$ be $\vee$-strategies for $\mathcal{E}$ and ρ be a pre-solution of $\mathcal{E}(\sigma)$. The $\vee$-strategy σ' is called *improvement of σ w.r.t. ρ* iff the following conditions are fulfilled:

(1) If $\rho \neq [\![\mathcal{E}]\!]\rho$, then $[\![\mathcal{E}(\sigma')]\!]\rho > \rho$.
(2) For all $\vee$-expressions $e_1 \vee \cdots \vee e_k$ occurring in $\mathcal{E}$ the following holds:
If $\sigma'(e) \neq \sigma(e)$, then $[\![\sigma'(e)]\!]\rho > [\![\sigma(e)]\!]\rho$. □

We can now formulate the $\vee$-strategy improvement algorithm for computing least solutions of systems of monotone equations over complete linearly ordered sets. The input is a system $\mathcal{E}$ of monotone equations over a complete linearly ordered set, a $\vee$-strategy σ_{init} for $\mathcal{E}$, and a pre-solution ρ_{init} of $\mathcal{E}(\sigma_{\text{init}})$. In order to compute the *least* and not just some solution, we additionally require that $\rho_{\text{init}} \leq \mu[\![\mathcal{E}]\!]$ holds:

Algorithm 2 The $\vee$-Strategy Improvement Algorithm

Input : - A system $\mathcal{E}$ of monotone equations over a complete linearly ordered set
- A $\vee$-strategy σ_{init} for $\mathcal{E}$
- A pre-solution ρ_{init} of $\mathcal{E}(\sigma_{\text{init}})$ with $\rho_{\text{init}} \leq \mu[\![\mathcal{E}]\!]$

Output : The least solution $\mu[\![\mathcal{E}]\!]$ of $\mathcal{E}$

$\sigma \leftarrow \sigma_{\text{init}}$;
$\rho \leftarrow \rho_{\text{init}}$;
while $(\rho \neq [\![\mathcal{E}]\!]\rho)$ {
  $\sigma \leftarrow$ improvement of σ w.r.t. ρ;
  $\rho \leftarrow \mu_{\geq\rho}[\![\mathcal{E}(\sigma)]\!]$;
}
return ρ;

Lemma 24. *Let $\mathcal{E}$ be a system of monotone equations over a complete linearly ordered set. For $i \in \mathbb{N}$, let ρ_i be the value of the program variable ρ and σ_i be the value of the program variable σ in the $\vee$-strategy improvement algorithm (Algorithm 2) after the i-th evaluation of the loop-body. The following statements hold for all $i \in \mathbb{N}$:*

(1) $\rho_i \leq \mu[\![\mathcal{E}]\!]$.
(2) $\rho_i \leq [\![\mathcal{E}(\sigma_{i+1})]\!]\rho_i$.
(3) $\rho_{i+1} = \mu_{\geq \rho_i}[\![\mathcal{E}(\sigma_{i+1})]\!]$.
(4) *If* $\rho_i < \mu[\![\mathcal{E}]\!]$, *then* $\rho_{i+1} > \rho_i$.
(5) *If* $\rho_i = \mu[\![\mathcal{E}]\!]$, *then* $\rho_{i+1} = \rho_i$.
(6) *Whenever the $\vee$-strategy improvement algorithm terminates, it computes the least solution $\mu[\![\mathcal{E}]\!]$ of $\mathcal{E}$.* □

Now, assume that $\mathcal{E}$ is a system of $\vee$-cmcave equations. In this case our $\vee$-strategy improvement algorithm terminates and returns the least solution at the latest after considering each $\vee$-strategy at most $|\mathbf{X}|$ times. For simplicity, we assume w.l.o.g. that each equation of $\mathcal{E}$ is of the form $\mathbf{x} = -\infty \vee e$. Then, we start our $\vee$-strategy improvement algorithm with a $\vee$-strategy σ_{init} such that $\mathcal{E}(\sigma_{\text{init}}) = \{\mathbf{x} = -\infty \mid \mathbf{x} \in \mathbf{X}\}$ and the pre-solution $\underline{-\infty}$ of $\mathcal{E}(\sigma_{\text{init}})$. Using the notations from Lemma 24, for all $i \in \mathbb{N}$, the value $\rho_{i+1} = \mu_{\geq \rho_i}[\![\mathcal{E}(\sigma_{i+1})]\!]$ can be determined as follows:

Lemma 25. *Let*

$$\mathbf{X}^{-\infty} := \{\mathbf{x} \in \mathbf{X} \mid \mathbf{x} = e \in \mathcal{E}(\sigma_{i+1}) \text{ and } [\![e]\!]\rho_i = -\infty\} \tag{105}$$

$$\mathbf{X}^{\infty} := \{\mathbf{x} \in \mathbf{X} \mid \mathbf{x} = e \in \mathcal{E}(\sigma_{i+1}) \text{ and } [\![e]\!]\rho_i = \infty\} \tag{106}$$

$$\mathbf{X}' := \mathbf{X} \setminus (\mathbf{X}^{-\infty} \cup \mathbf{X}^{\infty}) \tag{107}$$

$$\mathcal{E}' := \{\mathbf{x} = e \in \mathcal{E}(\sigma_{i+1}) \mid \mathbf{x} \in \mathbf{X}'\}[-\infty/\mathbf{X}^{-\infty}][\infty/\mathbf{X}^{\infty}]. \tag{108}$$

Here, $\mathcal{E}'$ denotes the system of cmcave equations that is obtained from $\mathcal{E}(\sigma_{i+1})$ by removing all equations $\mathbf{x} = e$ with $\mathbf{x} \notin \mathbf{X}'$ and then replacing all occurrences of variables from $\mathbf{X}^{-\infty}$ in the right-hand sides with the constant $-\infty$ and all occurrences of variables from $\mathbf{X}^{\infty}$ in the right-hand sides with the constant ∞. Then, for all $\mathbf{x}' \in \mathbf{X}'$,

$$\begin{aligned}\rho_{i+1}(\mathbf{x}') &= \mu_{\geq \rho_i}[\![\mathcal{E}(\sigma_{i+1})]\!](\mathbf{x}') \\ &= \mu_{\geq \rho_i|_{\mathbf{X}'}}[\![\mathcal{E}']\!](\mathbf{x}') \\ &= \sup\{\rho(\mathbf{x}') \mid \rho : \mathbf{X}' \to \mathbb{R} \text{ and } \rho(\mathbf{x}) \leq [\![e]\!]\rho \text{ for all equations } \mathbf{x} = e \in \mathcal{E}'\}.\end{aligned}$$

Further, $\rho_{i+1}(\mathbf{x}^{-\infty}) = \mu_{\geq \rho_i}[\![\mathcal{E}(\sigma_{i+1})]\!](\mathbf{x}^{-\infty}) = -\infty$ for all $\mathbf{x}^{-\infty} \in \mathbf{X}^{-\infty}$, and $\rho_{i+1}(\mathbf{x}^{\infty}) = \mu_{\geq \rho_i}[\![\mathcal{E}(\sigma_{i+1})]\!](\mathbf{x}^{\infty}) = \infty$ for all $\mathbf{x}^{\infty} \in \mathbf{X}^{\infty}$,

Provided that $\mathcal{E}$ is a system of $\vee$-cmcave equations, ρ_{i+1} can be computed by solving $|\mathbf{X}|$ convex optimization problems. Moreover, ρ_{i+1} is uniquely determined through the system $\mathcal{E}$, the $\vee$-strategy σ_{i+1} and the set $\mathbf{X}^{\infty}$ of all variables that are already known to be ∞. □

The sequence $((\rho_i, \{\mathbf{x} \in \mathbf{X} \mid \mathbf{x} = e \in \mathcal{E}(\sigma_{i+1}) \text{ and } [\![e]\!]\rho_i = \infty\}))_i$ is strictly increasing (ordered component-wise), since the sequence $(\rho_i)_i$ is strictly increasing and the sequence $(\{\mathbf{x} \in \mathbf{X} \mid \mathbf{x} = e \in \mathcal{E}(\sigma_{i+1}) \text{ and } [\![e]\!]\rho_i = \infty\})_i$ is increasing. By Lemma 25, ρ_{i+1} is uniquely determined through the system $\mathcal{E}$, the $\vee$-strategy σ_{i+1} and the set $\{\mathbf{x} \in \mathbf{X} \mid \mathbf{x} = e \in \mathcal{E}(\sigma_{i+1}) \text{ and } [\![e]\!]\rho_i = \infty\}$. Therefore, the $\vee$-strategy improvement algorithm considers each $\vee$-strategy at most $|\mathbf{X}|$ times (considering some $\vee$-strategy more than $|\mathbf{X}|$ times would contradict the fact that $(\rho_i)_i$ is strictly increasing). We get:

Theorem 26 (*Gawlitza and Seidl, 2010*)**.** *Let $\mathcal{E}$ be a system of $\vee$-cmcave equations. The $\vee$-strategy improvement algorithm (Algorithm 2) computes the least solution $\mu[\![\mathcal{E}]\!]$ of $\mathcal{E}$ and performs at most $|\Sigma| \cdot |\mathbf{X}|$ $\vee$-strategy improvement steps. If $\mathcal{E}$ is a system of $\vee$-cmcave equations, at most $|\mathbf{X}|$ convex optimization problems must be solved for every $\vee$-strategy improvement step.* □

Example 27. We consider the system

$$\mathcal{E} = \left\{ \mathbf{x} = -\infty \vee \tfrac{1}{2} \vee \sqrt{\mathbf{x}} \vee \tfrac{7}{8} + \sqrt{\mathbf{x} - \tfrac{47}{64}} \right\} \tag{109}$$

of $\vee$-cmcave equations. We start with the uniquely determined $\vee$-strategy σ_0 such that

$$\mathcal{E}(\sigma_0) = \{\mathbf{x} = -\infty\} \tag{110}$$

and with the solution $\rho_0 := \{\mathbf{x} \mapsto -\infty\}$ of $\mathcal{E}(\sigma_0)$. Since $\rho_0 \neq [\![\mathcal{E}]\!]\rho_0$, we improve the $\vee$-strategy σ_0 w.r.t. ρ_0 to a $\vee$-strategy σ_1 . Necessarily, we get

$$\mathcal{E}(\sigma_1) = \left\{ \mathbf{x} = \frac{1}{2} \right\}. \tag{111}$$

By Lemma 25, we get

$$\rho_1(\mathbf{x}) = \mu_{\geq \rho_0}[\![\mathcal{E}(\sigma_1)]\!](\mathbf{x}) = \sup \left\{ \mathbf{x} \mid \mathbf{x} \leq \frac{1}{2} \right\} = \frac{1}{2}. \tag{112}$$

Since $\sqrt{\frac{1}{2}} > \frac{1}{2}$ and $\frac{7}{8} + \sqrt{\frac{1}{2} - \frac{47}{64}} < \frac{1}{2}$, we necessarily improve the $\vee$-strategy σ_1 w.r.t. ρ_1 to the uniquely determined $\vee$-strategy σ_2 such that

$$\mathcal{E}(\sigma_2) = \left\{ \mathbf{x} = \sqrt{\mathbf{x}} \right\}. \tag{113}$$

Again by Lemma 25, we get

$$\rho_2(\mathbf{x}) = \mu_{\geq \rho_1}[\![\mathcal{E}(\sigma_2)]\!](\mathbf{x}) = \sup \left\{ \mathbf{x} \mid \mathbf{x} \leq \sqrt{\mathbf{x}} \right\} = 1. \tag{114}$$

Since

$$\frac{7}{8} + \sqrt{1 - \frac{47}{64}} > \frac{7}{8} + \sqrt{1 - \frac{60}{64}} = \frac{9}{8} > 1, \tag{115}$$

we get

$$\mathcal{E}(\sigma_3) = \left\{ \mathbf{x} = \frac{7}{8} + \sqrt{\mathbf{x} - \frac{47}{64}} \right\}. \tag{116}$$

Again by Lemma 25, we get

$$\rho_3(\mathbf{x}) = \mu_{\geq \rho_2}[\![\mathcal{E}(\sigma_3)]\!](\mathbf{x}) = \sup \left\{ \mathbf{x} \mid \mathbf{x} \leq \frac{7}{8} + \sqrt{\mathbf{x} - \frac{47}{64}} \right\} = 2. \tag{117}$$

Thus, we finally have $\rho_3 = \{\mathbf{x} \mapsto 2\}$. The algorithm terminates, because ρ_3 solves $\mathcal{E}$. Thus, $\rho_3 = \mu[\![\mathcal{E}]\!]$. We have found the least solution. For each $\vee$-strategy improvement step, we solved convex quadratic optimization problems that can be solved through semi-definite programming. □

6.3. The harmonic oscillator

We continue with Example 12. After introducing $-\infty$ at the right-hand sides, we obtain the following system of $\vee$-cmcave equations:

$$\begin{aligned}
\mathbf{x}_{\mathbf{st},p_1} &= -\infty \vee 0 \vee ([\![s]\!]^{\mathcal{R}}\{p \mapsto \mathbf{x}_{\mathbf{st},p} \mid p \in P\})(p_1) \\
\mathbf{x}_{\mathbf{st},p_2} &= -\infty \vee 1 \vee ([\![s]\!]^{\mathcal{R}}\{p \mapsto \mathbf{x}_{\mathbf{st},p} \mid p \in P\})(p_2) \\
\mathbf{x}_{\mathbf{st},p_3} &= -\infty \vee 0 \vee ([\![s]\!]^{\mathcal{R}}\{p \mapsto \mathbf{x}_{\mathbf{st},p} \mid p \in P\})(p_3) \\
\mathbf{x}_{\mathbf{st},p_4} &= -\infty \vee 1 \vee ([\![s]\!]^{\mathcal{R}}\{p \mapsto \mathbf{x}_{\mathbf{st},p} \mid p \in P\})(p_4) \\
\mathbf{x}_{\mathbf{st},p_5} &= -\infty \vee 7 \vee ([\![s]\!]^{\mathcal{R}}\{p \mapsto \mathbf{x}_{\mathbf{st},p} \mid p \in P\})(p_5)
\end{aligned} \tag{118}$$

In this example we have $3^5 = 243$ different $\vee$-strategies. It is clear that the algorithm will switch to the $\vee$-strategy that is given by the finite constants in the first step. At each equation, it then can switch to the non-constant expression, but then, because it constructs a strictly increasing sequence, it will never return to the constant. Summarizing, the $\vee$-strategy improvement algorithm performs at most 6 $\vee$-strategy improvement steps. In fact our proof-of-concept implementation performs 4 $\vee$-strategy improvement steps when solving this example. The last $\vee$-strategy that the algorithm considers leads to the system

$$\begin{aligned}
\mathbf{x}_{\mathbf{st},p_1} &= ([\![s]\!]^{\mathcal{R}}\{p \mapsto \mathbf{x}_{\mathbf{st},p} \mid p \in P\})(p_1)\\
\mathbf{x}_{\mathbf{st},p_2} &= ([\![s]\!]^{\mathcal{R}}\{p \mapsto \mathbf{x}_{\mathbf{st},p} \mid p \in P\})(p_2)\\
\mathbf{x}_{\mathbf{st},p_3} &= ([\![s]\!]^{\mathcal{R}}\{p \mapsto \mathbf{x}_{\mathbf{st},p} \mid p \in P\})(p_3)\\
\mathbf{x}_{\mathbf{st},p_4} &= ([\![s]\!]^{\mathcal{R}}\{p \mapsto \mathbf{x}_{\mathbf{st},p} \mid p \in P\})(p_4)\\
\mathbf{x}_{\mathbf{st},p_5} &= 7
\end{aligned} \tag{119}$$

of cmcave equations. The current approximate at this point of time does not assign ∞ to any variable. Because of Lemma 25, in order to determine the next value for the variable $\mathbf{x}_{\mathbf{st},p_k}$ (for $k \in \{1,\dots,5\}$), we solve the following convex optimization problem

$$\begin{aligned}
&\sup\{\rho(\mathbf{x}_{\mathbf{st},p_k}) \mid \rho : \mathbf{X} \to \mathbb{R},\ \mathbf{x}_{\mathbf{st},p_5} \le 7,\\
&\qquad \rho(\mathbf{x}_{\mathbf{st},p_i}) \le ([\![s]\!]^{\mathcal{R}}\{q \mapsto \rho(\mathbf{x}_{\mathbf{st},q}) \mid q \in P\})(p_i) \text{ for all } i \in \{1,\dots,4\}\}.
\end{aligned} \tag{120}$$

For that, we solve the following semi-definite programming problem that is obtained from the convex optimization problem through unfolding the definition of the relaxed abstract semantics:

$$\sup\ \mathbf{x}_{\mathbf{st},p_k} \tag{121}$$

$$\mathbf{x}_{\mathbf{st},p_1} \le C_k \bullet X^{(1)} \qquad X^{(1)} \succeq 0 \qquad X^{(1)}_{1\cdot 1} = 1 \tag{122}$$

$$B_1 \bullet X^{(1)} \le \mathbf{x}_{\mathbf{st},p_1} \qquad \cdots \qquad B_5 \bullet X^{(1)} \le \mathbf{x}_{\mathbf{st},p_5} \tag{123}$$

$$\mathbf{x}_{\mathbf{st},p_2} \le C_k \bullet X^{(2)} \qquad X^{(2)} \succeq 0 \qquad X^{(2)}_{1\cdot 1} = 1 \tag{124}$$

$$B_1 \bullet X^{(2)} \le \mathbf{x}_{\mathbf{st},p_1} \qquad \cdots \qquad B_5 \bullet X^{(2)} \le \mathbf{x}_{\mathbf{st},p_5} \tag{125}$$

$$\mathbf{x}_{\mathbf{st},p_3} \le C_k \bullet X^{(3)} \qquad X^{(3)} \succeq 0 \qquad X^{(3)}_{1\cdot 1} = 1 \tag{126}$$

$$B_1 \bullet X^{(3)} \le \mathbf{x}_{\mathbf{st},p_1} \qquad \cdots \qquad B_5 \bullet X^{(3)} \le \mathbf{x}_{\mathbf{st},p_5} \tag{127}$$

$$\mathbf{x}_{\mathbf{st},p_4} \le C_k \bullet X^{(4)} \qquad X^{(4)} \succeq 0 \qquad X^{(4)}_{1\cdot 1} = 1 \tag{128}$$

$$B_1 \bullet X^{(4)} \le \mathbf{x}_{\mathbf{st},p_1} \qquad \cdots \qquad B_5 \bullet X^{(4)} \le \mathbf{x}_{\mathbf{st},p_5} \tag{129}$$

$$\mathbf{x}_{\mathbf{st},p_5} \le 7 \tag{130}$$

The matrices $B_1,\dots,B_5,C_1,\dots,C_5$ are defined in Example 12. Solving the above semi-definite programming problem gives us the final values for the variables $\mathbf{x}_{\mathbf{st},p_1},\dots,\mathbf{x}_{\mathbf{st},p_5}$. We get

$$\begin{aligned}
\mu[\![\mathcal{E}]\!] = \{&\mathbf{x}_{\mathbf{st},p_1} \mapsto 1.8708..,\ \mathbf{x}_{\mathbf{st},p_2} \mapsto 1.8708..,\\
&\mathbf{x}_{\mathbf{st},p_3} \mapsto 1.5275..,\ \mathbf{x}_{\mathbf{st},p_4} \mapsto 1.5275..,\ \mathbf{x}_{\mathbf{st},p_5} \mapsto 7\}.
\end{aligned} \tag{131}$$

Hence, the following invariants hold at program point **st** of the harmonic oscillator:

$$-x_1 \le 1.8708 \qquad x_1 \le 1.8708 \qquad -x_2 \le 1.5275 \tag{132}$$

$$x_2 \le 1.5275 \qquad 2x_1^2 + 3x_2^2 + 2x_1x_2 \le 7. \tag{133}$$

For this example the $\vee$-strategy improvement algorithm presented in this section finds the same invariants as the $\wedge$-strategy improvement algorithm presented in Section 5 (cf. Section 5.5).

7. Comparison and conclusion

We have discussed how strategy iteration can be used for solving systems of $\vee$-cmcave equations. In the context of static program analysis, such equation systems are useful for approximating the abstract semantics of programs w.r.t. quadratic templates through semi-definite relaxations. We discussed two different strategy improvement approaches for solving systems of $\vee$-cmcave equations:

The $\wedge$-strategy iteration approach of Adjé et al. (2010) successively approximates the given equation system by systems of affine inequalities which can be efficiently solved by linear programming. The resulting method works similar to Newton's method. For each approximate, an improved $\wedge$-strategy (a system of affine inequalities) can be efficiently determined through semi-definite programming.

As an alternative approach, we discussed the $\vee$-strategy improvement approach of Gawlitza and Seidl (2007a,b). From an algorithmic perspective, this approach differs significantly from the $\wedge$-strategy improvement approach. $\vee$-strategy iteration, when applied to quadratic zones, in each iteration combines one constraint for each program point and polynomial template into a *global* semi-definite programming problem which is jointly solved.

The advantage of the $\vee$-strategy iteration approach is that (given an ideal SDP solver) the number of iterations is guaranteed to be finite and that it guarantees minimality of the obtained solution. The draw-back, however, is that only after termination, a safe invariant is found. Intermediate approximates to the least solution are not safe.

The $\wedge$-strategy iteration approach on the other hand, when applied to quadratic templates, relies on solving (dual) SDP problems *locally* for every constraint separately — each of which typically involves just few unknowns of the analysis problem. The *global* task of determining the next approximate for all program points and polynomial templates then is delegated to linear programming. The disadvantage of the $\wedge$-strategy iteration approach is that the iteration is not guaranteed to *terminate* but only to *converge* to a solution. Moreover, this solution is not necessarily minimal. On the other hand (again assuming ideal solvers for semi-definite and linear programming), it produces a decreasing sequence of post-fixpoints. Thus, the iteration may any time be terminated with a valid program invariant. Also, the speed of convergence is – as for Newton's method – usually quite good. Another advantage of the $\wedge$-strategy iteration approach is that LP solvers scale to larger problems than SDP solvers. Therefore, we expect the $\wedge$-strategy iteration approach to be applicable not just to small, but also to medium sized inputs. A detailed practical comparison w.r.t. efficiency and precision, however, remains for future work.

References

Adjé, A., Gaubert, S., Goubault, E., 2008. Computing the smallest fixed point of nonexpansive mappings arising in game theory and static analysis of programs. In: Proceedings of the Eighteenth International Symposium on Mathematical Theory of Networks and Systems (MTNS2008).

Adjé, A., Gaubert, S., Goubault, E., 2010. Coupling policy iteration with semi-definite relaxation to compute accurate numerical invariants in static analysis. In: Gordon, A.D. (Ed.), ESOP. In: LNCS, vol. 6012. Springer, pp. 23–42.

Borchers, B., 1999a. CSDP 2.3 user's guide. Optimization Methods and Software 11, 597–611.

Borchers, B., 1999b. CSDP, A C library for semidefinite programming. Optimization Methods and Software 11, 613.

Borchers, B., Young, J., 2007. Implementation of a primal–dual method for SDP on a shared memory parallel architecture. Computational Optimization and Applications 37, 355–369. doi:10.1007/s10589-007-9030-3.

Boyd, S., Vandenberghe, L., 2004. Convex Optimization. Cambridge University Press.

Costan, A., Gaubert, S., Goubault, E., Martel, M., Putot, S., 2005. A Policy Iteration Algorithm for Computing Fixed Points in Static Analysis of Programs.. In: Computer Aided Verification, 17th Int. Conf. (CAV). In: LNCS, vol. 3576. Springer Verlag, pp. 462–475.

Cousot, P., Cousot, R., 1977. Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints. In: 4th ACM SIGPLAN-SIGACT Symp. on Principles of Programming Languages (POPL). pp. 238–252.

Feron, E., Alegre, F., 2008a. Control software analysis, Part I Open-loop properties. CoRR abs/0809.4812.

Feron, E., Alegre, F., 2008b. Control software analysis, Part II: Closed-loop analysis. CoRR abs/0812.1986.

Fukuda, M., Braams, B.J., Nakata, M., Overton, M.L., Percus, J.K., Yamashita, M., Zhao, Z., 2007. Large-scale semidefinite programs in electronic structure calculation. Mathematical Programming 109 (2–3), 553–580.

Gaubert, S., Goubault, E., Taly, A., Zennou, S., 2007. Static analysis by policy iteration on relational domains. In: Nicola (2007), pp. 237–252.

Gawlitza, T., Seidl, H., 2007a. Precise fixpoint computation through strategy iteration. In: Nicola (2007), pp. 300–315.
Gawlitza, T., Seidl, H., 2007b. Precise relational invariants through strategy iteration. In: Duparc, J., Henzinger, T.A. (Eds.), CSL. In: LNCS, vol. 4646. Springer, pp. 23–40.
Gawlitza, T., Seidl, H., 2008. Precise interval analysis vs. parity games. In: Cuéllar, J., Maibaum, T. S.E., Sere, K. (Eds.), FM. In: LNCS, vol. 5014. Springer, pp. 342–357.
Gawlitza, T.M., Seidl, H., 2010. Computing relaxed abstract semantics w.r.t. quadratic zones precisely. In: Cousot, R., Martel, M. (Eds.), SAS. In: Lecture Notes in Computer Science, vol. 6337. Springer, pp. 271–286.
Gawlitza, T.M., Seidl, H., Solving systems of rational equations through strategy iteration. TOPLAS (in press).
Miné, A., 2001. The octagon abstract domain. In: WCRE. pp. 310.
Nemirovski, A., 2005. Modern Convex Optimization. Department ISYE, Georgia Institute of Technology.
Nicola, R.D. (Ed.), 2007. Programming Languages and Systems, 16th European Symposium on Programming, ESOP 2007, Held as Part of the Joint European Conferences on Theory and Practics of Software, ETAPS 2007, Braga, Portugal, March 24 - April 1, 2007, Proceedings. In: ESOP. In: Lecture Notes in Computer Science, vol. 4421. Springer.
Ortega, J., Rheinboldt, W., 1970. Iterative Solution of Nonlinear Equations in Several Variables. Academic Press.
Sankaranarayanan, S., Colón, M., Sipma, H.B., Manna, Z., 2006. Efficient strongly relational polyhedral analysis. In: Emerson, E.A., Namjoshi, K.S. (Eds.), VMCAI. In: Lecture Notes in Computer Science, vol. 3855. Springer, pp. 111–125.
Sankaranarayanan, S., Sipma, H.B., Manna, Z., 2005. Scalable analysis of linear systems using mathematical programming. In: Cousot, R. (Ed.), VMCAI. In: LNCS, vol. 3385. Springer, pp. 25–41.
Schrijver, A., 1986. Theory of Linear and Integer Programming. Wiley.
Sturm, J.F., 1999. Using sedumi 1.02, a matlab toolbox for optimization over symmetric cones. Optimization Methods and Software 11–12, 625–653.
Todd, M.J., 2001. Semidefinite optimization. Acta Numerica 10, 515–560.
Toh, K., Todd, M.J., Tütüncü, R., 1999. SDPT3 — A Matlab software package for semidefinite programming, Version 1.3. In: Optimization Methods and Software, vol. 11. Taylor and Francis, pp. 545–581.
Tütüncü, R.H., Toh, K.C., Todd, M.J., 2003. Solving semidefinite-quadratic-linear programs using SDPT3. Mathematical Programming 95 (2), 189–217.
Vavasis, S.A., 1990. Quadratic programming is in NP. Information Processing Letters 36 (2), 73–77.
Wright, S.J., 1997. Primal–dual Interior-point Methods. SIAM.
Yamashita, M., Fujisawa, K., Kojima, M., 2003a. Implementation and evaluation of SDPA 6.0 (semidefinite programming algorithm 6.0). Optimization Methods and Software 18, 491–505.
Yamashita, M., Fujisawa, K., Kojima, M., 2003b. SDPARA: SemiDefinite Programming Algorithm paRAllel version. Parallel Computing 29 (8), 1053–1067.