A Lattice-Theoretical View of Strategy Iteration

Paolo Baldan
University of Padova, Italy

Richard Eggert
Universität Duisburg-Essen, Germany

Barbara König
Universität Duisburg-Essen, Germany

Tommaso Padoan
University of Padova, Italy

Abstract

Strategy iteration is a technique frequently used for two-player games in order to determine the winner or compute payoffs, but to the best of our knowledge no general framework for strategy iteration has been considered. Inspired by previous work on simple stochastic games, we propose a general formalisation of strategy iteration for solving least fixpoint equations over a suitable class of complete lattices, based on MV-chains. We devise algorithms that can be used for non-expansive fixpoint functions represented as so-called min- respectively max-decompositions. Correspondingly, we develop two different techniques: strategy iteration from above, which has to solve the problem that iteration might reach a fixpoint that is not the least, and from below, which is algorithmically simpler, but requires a more involved correctness argument. We apply our method to solve energy games and compute behavioural metrics for probabilistic automata.

2012 ACM Subject Classification Theory of computation → Program verification; Theory of computation → Solution concepts in game theory

Keywords and phrases games, strategy iteration, fixpoints, energy games, behavioural metrics

Digital Object Identifier 10.4230/LIPIcs.CSL.2023.7

Related Version *Full Version*: https://arxiv.org/abs/2207.09872 [7]

Funding Partially supported by the DFG project SpeQt and by the Ministero dell'Universtà e della Ricerca Scientifica of Italy, under Grant No. 201784YSZ5, PRIN2017 – ASPRA.

1 Introduction

Strategy iteration (or policy iteration) is a well known technique in computer science. It has been widely adopted for the solution of two-player games where the players, say Max and Min, aim at maximising and minimising, respectively, some payoff. In many cases there exists an optimal strategy for each player where no deviation is advisable as long as the other player plays optimally. We here assume a scenario where memoryless (or positional) strategies are sufficient. The general idea of strategy iteration is to iteratively fix a strategy for one player, compute the optimal answering strategy for the other player and then improve the strategy of the first player. As long as there are only finitely many strategies, an optimal strategy is bound to be found at some point. Such strategy iteration methods exist for Markov decision processes [21] and for a variety of games, such as simple stochastic games [13, 23, 1], (discounted) mean-payoff games [32, 10] and parity games [31, 28].

Similar ideas apply also to a wide range of different problems. For instance, the computation of behavioural distances for systems embodying quantitative information, e.g., time, probability or cost, is often based on some form of lifting of distances on states [2, 4, 8]. In turn the lifting relies on couplings which play the role of strategies and algorithms based on a progressive improvement of couplings have been devised [2, 3].

Motivating example. To help the intuition, we review simple stochastic games (SSGs) [13] and strategy iteration in that setting as discussed in [5, 6]. An SSG consists of a set of states V, partitioned in four subsets MIN, MAX, AV and $SINK$. States in $SINK$ (sink states) have no successor and yield a payoff in $[0,1]$. For states in AV (average states) the successor is determined by a probability distribution over V, i.e., intuitively, the environment makes a probabilistic choice. In a state in MIN, player Min chooses a successor trying to minimise the expected payoff, while in a state in MAX, it is the player Max that chooses, with the aim of maximising the expected payoff. An example of an SSG is in Fig. 1 on page 7.

When Min and Max play optimally, the expected payoff at each state is given by the least fixpoint of the function $\mathcal{V}\colon [0,1]^V \to [0,1]^V$, defined for $a\colon V \to [0,1]$ and $v \in V$ by

$$\mathcal{V}(a)(v) = \begin{cases} \max_{v \to v'} a(v') & v \in MAX \\ \min_{v \to v'} a(v') & v \in MIN \\ \sum_{v' \in V} p(v)(v') \cdot a(v') & v \in AV \\ c(v) & v \in SINK \end{cases}$$

with $p(v)(v')$ the probability of state v reaching v' and $c(v) \in [0,1]$ the payoff of sink state v.

The idea of strategy iteration from below, instantiated to this context, is to compute the least fixpoint $\mu\mathcal{V}$ via an iteration of the following kind:

1. Guess a strategy $\sigma\colon MAX \to V$ for player Max, i.e., fix a successor for states in MAX.
2. Compute the least fixpoint of $\mathcal{V}_\sigma : [0,1]^V \to [0,1]^V$, which is defined as $\mathcal{V}$ in all cases apart from $v \in MAX$, where we set $\mathcal{V}(a)(v) = a(\sigma(v))$. This fixpoint computation is simpler than the original one and it can be done efficiently via linear programming.
3. Based on $\mu\mathcal{V}_\sigma$, try to improve the strategy for Max. If the strategy does not change, we have computed a fixpoint of $\mathcal{V}$ and, since iteration is from below, this is necessarily the least fixpoint. If the strategy changes, continue with step 2.

A similar approach can be used for converging to the least fixpoint from above. In this case, it is now player Min who fixes a strategy which is progressively improved. This procedure is well-known to work for stopping games [13], i.e., SSGs where each combination of strategies ensures termination, since for these games $\mathcal{V}$ has a unique fixpoint. However, in general, when iterating from above the procedure may get stuck at some fixpoint which is not the least fixpoint of $\mathcal{V}$, a problem which is solved by the theory developed in [5] which can be used to "skip" this fixpoint and continue the iteration from there.

While, as explained above, the general idea of strategy iteration is used in many different settings, to the best of our knowledge a general definition of strategy iteration is still missing. The goal of the present paper is to provide a general and abstract formulation of an algorithm for strategy iteration, proved correct once and for all, which instantiates to a variety of problems. The key observation is that optimal strategies very often arise from some form of extremal (least or greatest) fixpoint of a suitable non-expansive function f over a complete MV-chain [27], the paradigmatic example being the real interval $[0,1]$ with the usual order. We propose a framework where the operation of fixing a strategy for one of the players is captured abstractly, in terms of so-called min- or max-decompositions of the function of interest. Then, we devise strategy iteration approaches which converge to the fixpoint of interest by successively improving the strategy for the chosen player. We will assume that the interest is in least fixpoints, but the theory can be dualised. We propose two strategy iteration algorithms that converge to the least fixpoint "from below" and "from above", respectively. As it happens for SSGs, in the latter case the iteration can reach a fixpoint

which is not the least. Clearly, whenever the function f of interest has a unique fixpoint this problem disappears. Moreover, in some cases, even though f has multiple fixpoints, it can be "patched" in a way that the modified function has the fixpoint of interest as its only fixpoint. Otherwise, we can rely on the results in [5] to check whether the reached fixpoint is the least one and whenever it is not, to get closer to the desired fixpoint and continue the iteration.

Strategy iteration approaches can be slow if compared to other algorithms, such as value iteration. However, the benefit of strategy iteration algorithms is that they allow an exact computation of the desired fixpoint, while other algorithms may never reach the sought-after extreme fixpoint but only converge towards it. This is the case, e.g., for simple stochastic games, where strategy iteration algorithms are the standard methods to obtain exact results. Additionally, strategy iteration, besides determining the fixpoint also singles out a strategy which allows one to obtain it, an information which is often of interest.

In summary, we propose the first, to the best of our knowledge, general definition of strategy iteration providing a lattice-theoretic formalisation of this technique. This requires to single out and solve in this general setting the fundamental challenges of these approaches, which already show up in earlier work on SSGs (see, e.g., [5, 10]). In the iteration from above, we may converge to a fixpoint that is not the least, while from below it is not straightforward to show that improving the strategy of Max leads to a larger fixpoint.

Known algorithms are rediscovered for SSGs and probabilistic automata [3]. Moreover new ones are obtained for energy games [9, 12] where movements in the game graph have an energy cost and the goal of one of the players is to avoid that the energy drops below zero. Given the number of different application domains where strategy iteration is or can be used, we feel that a general framework can unveil unexplored potentials. The two case studies (energy games and behavioural metrics) that we treat can be encoded into SSGs [3], but the obtained strategies have to be translated back to the original setting, which is not always trivial in general, and encodings usually come with a loss of efficiency. For instance, in order to solve SSGs a solver for linear programming is usually required, which is in general not needed for other applications.

The rest of the paper is structured as follows. In §2 we review some order-theoretic notions and recap some results from [5] for identifying least and greatest fixpoints. In §3 we devise two generalized strategy iteration algorithms, from above and from below, using SSGs (already treated in [6]) as a running example. In §4, we show how our technique applies to energy games, while in §5 we discuss an application to the computation of the behavioural distance for probabilistic automata.

Proofs and further material can be found in the full version of the paper [7].

2 Preliminaries on ordered structures and fixpoints

This section reviews some background used throughout the paper. This includes the basics of lattices and MV-algebras, where the functions of interest take values. We also recap some results from [5] useful for detecting if a fixpoint of a given function is the least (or greatest).

For X, Y sets, we denote by $\mathcal{P}(X)$ the powerset of X and $\mathcal{P}_{fin}(X)$ the set of finite subsets of X. Moreover, the set of functions from X to Y is denoted by either Y^X or $X \to Y$.

A partially ordered set $(P, \sqsubseteq)$ is often denoted simply as P, omitting the order relation. For a function $f : X \to P$, we will write $\arg\min_{x \in X} f(x)$ to denote the set of elements where f reaches the minimum, i.e., $\{x \in X \mid \forall y \in X.\, f(x) \sqsubseteq f(y)\}$ and, abusing the notation, we will write $z = \arg\min_{x \in X} f(x)$ instead of $z \in \arg\min_{x \in X} f(x)$.

The *join* and the *meet* of a subset $X \subseteq P$ (if they exist) are denoted $\bigsqcup X$ and $\bigsqcap X$.

A *complete lattice* is a partially ordered set $(\mathbb{L}, \sqsubseteq)$ such that each subset $X \subseteq \mathbb{L}$ admits a join $\bigsqcup X$ and a meet $\bigsqcap X$. A complete lattice $(\mathbb{L}, \sqsubseteq)$ always has a least element $\bot = \bigsqcap \mathbb{L}$ and a greatest element $\top = \bigsqcup \mathbb{L}$.

A function $f : \mathbb{L} \to \mathbb{L}$ is *monotone* if for all $l, l' \in \mathbb{L}$, if $l \sqsubseteq l'$ then $f(l) \sqsubseteq f(l')$. By Knaster-Tarski's theorem [29, Theorem 1], any monotone function on a complete lattice has a least fixpoint μf, characterised as the meet of all pre-fixpoints $\mu f = \bigsqcap\{l \mid f(l) \sqsubseteq l\}$ and, dually, a greatest fixpoint $\nu f = \bigsqcup\{l \mid l \sqsubseteq f(l)\}$, characterised as the join of all post-fixpoints. We denote by $Fix(f)$ the set of all fixpoints of f.

Given a set Y and a complete lattice $\mathbb{L}$, the set of functions $\mathbb{L}^Y = \{f \mid f : Y \to \mathbb{L}\}$, endowed with pointwise order, i.e., for $a, b \in \mathbb{L}^Y$, $a \sqsubseteq b$ if $a(y) \sqsubseteq b(y)$ for all $y \in Y$, is a complete lattice. We write $a \sqsubset b$ when $a \sqsubseteq b$ and $a \neq b$, i.e., for all $y \in Y$ we have $a(y) \sqsubseteq b(y)$ and $a(y) \sqsubset b(y)$ for some $y \in Y$.

We are also interested in the set of probability distributions $\mathcal{D}(Y) \subseteq [0,1]^Y$, i.e., functions $\beta : Y \to [0,1]$ such that $\sum_{y \in Y} \beta(y) = 1$.

An *MV-algebra* [27] is a tuple $\mathbb{M} = (M, \oplus, 0, \overline{(\cdot)})$ where $(M, \oplus, 0)$ is a commutative monoid and $\overline{(\cdot)} : M \to M$ maps each element to its *complement*, such that for all $x, y \in M$

1. $\overline{\overline{x}} = x$
2. $x \oplus \overline{0} = \overline{0}$
3. $\overline{(\overline{x} \oplus y)} \oplus y = \overline{(\overline{y} \oplus x)} \oplus x$.

We denote $1 = \overline{0}$ and subtraction $x \ominus y = \overline{\overline{x} \oplus y}$.

MV-algebras are endowed with a partial order, the so-called *natural order*, defined for $x, y \in M$, by $x \sqsubseteq y$ if $x \oplus z = y$ for some $z \in M$. When $\sqsubseteq$ is total, $\mathbb{M}$ is called an *MV-chain*. We will write $\mathbb{M}$ instead of M.

The natural order gives an MV-algebra a lattice structure where $\bot = 0$, $\top = 1$, $x \sqcup y = (x \ominus y) \oplus y$ and $x \sqcap y = \overline{\overline{x} \sqcup \overline{y}} = x \ominus (x \ominus y)$. We call the MV-algebra *complete* if it is a complete lattice, which is not true in general, e.g., $([0,1] \cap \mathbb{Q}, \leqslant)$.

▶ **Example 2.1.** A prototypical example of an MV-algebra is $([0,1], \oplus, 0, \overline{(\cdot)})$ where $x \oplus y = \min\{x + y, 1\}$, $\overline{x} = 1 - x$ and $x \ominus y = \max\{0, x - y\}$ for $x, y \in [0,1]$. The natural order is $\leqslant$ (less or equal) on the reals. Another example is $K = (\{0, \ldots, k\}, \oplus, 0, \overline{(\cdot)})$ where $n \oplus m = \min\{n + m, k\}$, $\overline{n} = k - n$ and $n \ominus m = \max\{n - m, 0\}$ for $n, m \in \{0, \ldots, k\}$. Both MV-algebras are complete and MV-chains.

We next briefly recap the theory from [5] which will be helpful in the paper for checking whether a fixpoint is the least or the greatest fixpoint of some underlying endo-function.

▶ Remark 2.2. Hereafter, unless stated otherwise, Y, Z will be assumed to be finite sets and $\mathbb{M}$ will be a complete MV-chain.

Given $a \in \mathbb{M}^Y$ we define its *norm* as $\|a\| = \max\{a(y) \mid y \in Y\}$. A function $f : \mathbb{M}^Y \to \mathbb{M}^Z$ is *non-expansive* if for all $a, b \in \mathbb{M}^Y$ it holds $\|f(b) \ominus f(a)\| \sqsubseteq \|b \ominus a\|$. It can be seen that non-expansive functions are monotone. A number of standard operators are non-expansive (e.g., constants, reindexing, max and min over a relation, average), and non-expansiveness is preserved by composition and disjoint union (see [5]). Given $Y' \subseteq Y$ and $\delta \in \mathbb{M}$, we write $\delta_{Y'}$ for the function defined by $\delta_{Y'}(y) = \delta$ if $y \in Y'$ and $\delta_{Y'}(y) = 0$, otherwise.

For a non-expansive endo-function $f : \mathbb{M}^Y \to \mathbb{M}^Y$ and $a \in \mathbb{M}^Y$, the theory in [5] provides a so-called a-approximation $f^a_\#$ of f, which is an endo-function over a suitable subset of Y. More precisely, define $[Y]^a = \{y \in Y \mid a(y) \neq 0\}$ and $\delta^a = \min\{a(y) \mid y \in [Y]^a\}$. For $0 \sqsubset \delta \in \mathbb{M}$ consider the functions $\alpha^{a,\delta} : \mathcal{P}([Y]^a) \to [a \ominus \delta, a]$ and $\gamma^{a,\delta} : [a \ominus \delta, a] \to \mathcal{P}([Y]^a)$, defined, for $Y' \in \mathcal{P}([Y]^a)$ and $b \in [a \ominus \delta, a]$, by

$$\alpha^{a,\delta}(Y') = a \ominus \delta_{Y'} \qquad \gamma^{a,\delta}(b) = \{y \in [Y]^a \mid a(y) \ominus b(y) \sqsupseteq \delta\}.$$

For a non-expansive function $f : \mathbb{M}^Y \to \mathbb{M}^Z$ and $\delta \in \mathbb{M}$, define $f^{a,\delta}_{\#} \colon \mathcal{P}([Y]^a) \to \mathcal{P}([Z]^{f(a)})$ as $f^{a,\delta}_{\#} = \gamma^{f(a),\delta} \circ f \circ \alpha^{a,\delta}$. The function $f^{a,\delta}_{\#}$ is antitone in the parameter δ and there exists a suitable value $\iota^a_f \sqsupset 0$, such that all functions $f^{a,\delta}_{\#}$ for $0 \sqsubset \delta \sqsubseteq \iota^a_f$ are equal. The function $f^a_{\#} := f^{a,\iota^a_f}_{\#}$ is called the *a-approximation* of f. When $\delta \sqsubseteq \delta_a$, the pair $\langle \alpha^{a,\delta}, \gamma^{a,\delta} \rangle$ is a Galois connection, a notion at the heart of abstract interpretation [14, 15], and $f^{a,\delta}_{\#} = \gamma^{f(a),\delta} \circ f \circ \alpha^{a,\delta}$ is the best correct approximation of f.

Intuitively, given some Y', the set $f^a_{\#}(Y')$ contains the points where a decrease of the values of a on the points in Y' "propagates" through the function f. The greatest fixpoint of $f^a_{\#}$ gives us the subset of Y where such a decrease is propagated in a cycle (so-called "vicious cycle"). Whenever $\nu f^a_{\#}$ is non-empty, one can argue that a cannot be the least fixpoint of f since we can decrease the value in all elements of $\nu f^a_{\#}$, obtaining a smaller prefixpoint. Interestingly, for non-expansive functions, it is shown in [5] that also the converse holds, i.e., emptiness of the greatest fixpoint of $f^a_{\#}$ implies that a is the least fixpoint.

▶ **Theorem 2.3** (soundness and completeness for fixpoints). *Let $\mathbb{M}$ be a complete MV-chain, Y a finite set and $f : \mathbb{M}^Y \to \mathbb{M}^Y$ be a non-expansive function. Let $a \in \mathbb{M}^Y$ be a fixpoint of f. Then $\nu f^a_{\#} = \varnothing$ if and only if $a = \mu f$.*

Using the above theorem we can check whether some fixpoint a of f is the least fixpoint. Whenever a is a fixpoint, but not yet the least fixpoint of f, it can be decreased by a fixed value in $\mathbb{M}$ (see [5] for the details) on the points in $\nu f^a_{\#}$ to obtain a smaller pre-fixpoint.

▶ **Lemma 2.4.** *Let $\mathbb{M}$ be a complete MV-chain, $f : \mathbb{M}^Y \to \mathbb{M}^Y$ a non-expansive function, $a \in \mathbb{M}^Y$ a fixpoint of f, and let $f^a_{\#}$ be the corresponding a-approximation. If a is not the least fixpoint and thus $\nu f^a_{\#} \neq \varnothing$ then there is $0 \sqsubset \delta \in \mathbb{M}$ such that $a \ominus \delta_{\nu f^a_{\#}}$ is a pre-fixpoint of f.*

In the following we will use this result as a "black box": we assume that given f and a fixpoint a of f we can determine whether $a = \mu f$ and, if not, obtain $a' \sqsubset a$ such $f(a') \sqsubseteq a'$.

The above theory can easily be dualised (see [5] for the details of the dual view).

3 Generalized strategy iteration

In this section we develop two strategy iteration techniques for determining least fixpoints. The first technique requires a so-called min-decomposition and approaches the least fixpoint from above, while the second uses a max-decomposition to ascend to the least fixpoint from below.

Hence fixpoint iteration from above is seen strictly from the point of view of the Min player, while fixpoint iteration from below is from the view of the Max player, who want to minimize respectively maximize the payoff. The player starts by guessing a strategy, which in the case of the Min (Max) player over-approximates (under-approximates) the true payoff. This strategy is then locally improved at each iteration based on the payoff produced by the player following such a strategy. That is, we compute fixpoints for a fixed strategy, which in a two-player game means that the opponent plays optimally. When the set of strategies is finite (or, at least, the search can be restricted to a finite set), an optimal strategy will be found at some point.

3.1 Function decomposition

We next introduce the setting where the generalisations of strategy iteration will be developed. We assume that the game we are interested in is played on a finite set of positions Y and the payoff at each position is an element of a suitable complete MV-chain $\mathbb{M}$. This payoff

is given by a function in $\mathbb{M}^Y$ that can be characterised as the least fixpoint of a monotone function $f : \mathbb{M}^Y \to \mathbb{M}^Y$. If we concentrate on the Min player, each position $y \in Y$ is assigned a set of functions $H_{\min}(y) \subseteq (\mathbb{M}^Y \to \mathbb{M})$ where each function $h \in H_{\min}(y)$ is one possible option that can be chosen by Min. Given $a \colon Y \to \mathbb{M}$ as the current estimate of the payoff, $h(a)$ is the resulting payoff at y. If the player does not have a choice, this set is a singleton. Since it is the aim of Min to minimise she will choose an h such that $h(a)$ is minimal.

▶ **Definition 3.1** (min-decomposition). *Let Y be a finite set and $\mathbb{M}$ be a complete MV-chain. Given a function $f : \mathbb{M}^Y \to \mathbb{M}^Y$, a* min-decomposition *of f is a function $H_{\min} : Y \to \mathcal{P}_{fin}(\mathbb{M}^Y \to \mathbb{M})$ such that for all $y \in Y$ the set $H_{\min}(y)$ consists only of monotone functions and for all $a \in \mathbb{M}^Y$ it holds $f(a)(y) = \min_{h \in H_{\min}(y)} h(a)$.*

Observe that any monotone function $f : \mathbb{M}^Y \to \mathbb{M}^Y$ admits a trivial min-decomposition I defined by $I(y) = \{h_y\}$ where $h_y(a) = f(a)(y)$ for all $a \in \mathbb{M}^Y$.

Whenever all $h \in H_{\min}(y)$ are not only monotone, but also non-expansive, it can be shown easily that f is also non-expansive and we can obtain an approximation as discussed in §2. Max-decompositions, with analogous properties, are defined dually, i.e. $H_{\max} : Y \to \mathcal{P}_{fin}(\mathbb{M}^Y \to \mathbb{M})$ and $f(a)(y) = \max_{h \in H_{\max}(y)} h(a)$.

Fixing a strategy can be seen as fixing, for all $y \in Y$, some element in $H_{\min}(y)$.

▶ **Definition 3.2** (strategy). *Let Y be a finite set, $\mathbb{M}$ be a complete MV-chain, $f : \mathbb{M}^Y \to \mathbb{M}^Y$ and let $H_{\min} : Y \to \mathcal{P}_{fin}(\mathbb{M}^Y \to \mathbb{M})$ be a min-decomposition of f. A* strategy *in $H_{\min}$ is a function $C : Y \to (\mathbb{M}^Y \to \mathbb{M})$ such that for all $y \in Y$ it holds that $C(y) \in H_{\min}(y)$. For a fixed C we define $f_C : \mathbb{M}^Y \to \mathbb{M}^Y$ as $f_C(a)(y) = C(y)(a)$ for all $a \in \mathbb{M}^Y$ and $y \in Y$.*

Strategies in a max-decomposition are defined dually.

The letter C stands for "choice" and typically μf_C is easier to compute than μf.

▶ **Example 3.3.** As a running example for illustrating our theory and the resulting algorithms we will use *simple stochastic games* (SSGs). We first show that they fall into the framework. Fix an SSG with a finite set V of states, partitioned into MIN, MAX, AV (average) and $SINK$. Successors of MIN and MAX states are given by a relation $\to \subseteq (MIN \cup MAX) \times V$, while $p : AV \to [0,1]^V$ maps each $v \in AV$ to a distribution $p(v) \in \mathcal{D}(V)$. Finally, $c : SINK \to [0,1]$ provides the payoff of sink states.

The fixpoint function $\mathcal{V} : [0,1]^V \to [0,1]^V$, as defined in the introduction, admits a min-decomposition $H_{\min} \colon V \to \mathcal{P}_{fin}(\mathbb{M}^V \to \mathbb{M})$ defined for all $a \in \mathbb{M}^Y$ as follows:

- for $v \in MIN$, $H_{\min}(v) = \{h_{v'} \mid v \to v'\}$ with $h_{v'}(a) = a(v')$;
- for $v \in MAX$, $H_{\min}(v) = \{h\}$ with $h(a) = \max_{v \to v'} a(v')$;
- for $v \in AV$, $H_{\min}(v) = \{h\}$ with $h(a) = \sum_{v' \in V} p(v)(v') \cdot a(v')$;
- for $v \in SINK$, $H_{\min}(v) = \{h\}$ with $h(a) = c(v)$.

A max-decomposition can be defined dually.

For instance, consider the SSG in Fig. 1 where $V = \{\mathbf{1}, \boldsymbol{\varepsilon}, \boldsymbol{av}, \boldsymbol{max}, \boldsymbol{min}\}$ with the obvious partitioning. The fixpoint function is $\mathcal{V} \colon [0,1]^V \to [0,1]^V$ defined, for $a \in [0,1]^V$, by

$$\mathcal{V}(a)(\mathbf{1}) = 1 \qquad \mathcal{V}(a)(\boldsymbol{\varepsilon}) = \varepsilon \qquad \mathcal{V}(a)(\boldsymbol{av}) = \frac{1}{2}a(\boldsymbol{min}) + \frac{1}{2}a(\boldsymbol{max})$$
$$\mathcal{V}(a)(\boldsymbol{max}) = \max\{a(\boldsymbol{\varepsilon}), a(\boldsymbol{av})\} \qquad \mathcal{V}(a)(\boldsymbol{min}) = \min\{a(\mathbf{1}), a(\boldsymbol{av})\}.$$

The min-decomposition defined in general above, in this case is $H_{\min} \colon V \to \mathcal{P}_{fin}(\mathbb{M}^V \to \mathbb{M})$ defined for all $a \in \mathbb{M}^Y$ as follows: for $v \in V \setminus \{\boldsymbol{min}\}$, we let $H_{\min}(v) = \{h\}$ with $h(a) = \mathcal{V}(a)(v)$, while $H_{\min}(\boldsymbol{min}) = \{h_{\mathbf{1}}, h_{\boldsymbol{av}}\}$ with $h_{\mathbf{1}}(a) = a(\mathbf{1})$ and $h_{\boldsymbol{av}}(a) = a(\boldsymbol{av})$. All strategies in

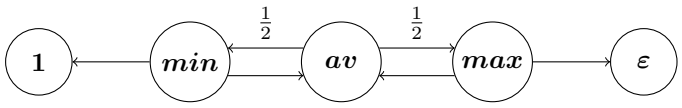


Figure 1 An example of a simple stochastic game. States $\mathbf{1}$, $\boldsymbol{\varepsilon}$ have payoff 1, $\varepsilon > 0$ respectively.

$H_{\min}$ assign to every state $v \in V\backslash\{\boldsymbol{min}\}$ the only element in $H_{\min}(v)$. Hence they are determined by the value on state $\boldsymbol{min}$: thus there are two strategies $C_1^{\min}, C_2^{\min}$ in $H_{\min}$ with $C_1^{\min}(\boldsymbol{min}) = h_{\mathbf{1}}$ and $C_2^{\min}(\boldsymbol{min}) = h_{\boldsymbol{av}}$.

Dually, a max-decomposition $H_{\max} \colon V \to \mathcal{P}_{fin}(\mathbb{M}^V \to \mathbb{M})$ is defined for all $a \in \mathbb{M}^Y$ as follows: $H_{\max}(v) = \{h\}$ with $h(a) = \mathcal{V}(a)(v)$ for all $v \in V\backslash\{\boldsymbol{max}\}$ and $H_{\max}(\boldsymbol{max}) = \{h_{\boldsymbol{\varepsilon}}, h_{\boldsymbol{av}}\}$ with $h_{\boldsymbol{\varepsilon}}(a) = a(\boldsymbol{\varepsilon})$ and $h_{\boldsymbol{av}}(a) = a(\boldsymbol{av})$. Again, there are two strategies $C_1^{\max}$ and $C_2^{\max}$ in $H_{\max}$ that differ for the value assigned to $\boldsymbol{max}$: $C_1^{\max}(\boldsymbol{max}) = h_{\boldsymbol{\varepsilon}}$ and $C_2^{\max}(\boldsymbol{max}) = h_{\boldsymbol{av}}$.

3.2 Strategy iteration from above

In this section we propose a generalized strategy iteration algorithm from above. It is based on a min-decomposition of the function and, intuitively, at each iteration the player Min improves her strategy. An issue here is that this iteration may get stuck at a fixpoint strictly larger than the least one. Recognising and overcoming this problem, thus continuing the iteration until the least fixpoint is reached, requires the theory described in §2.

The basic result that motivates strategy iteration from above is a characterisation of the least fixpoint of a function in terms of a min-decomposition.

▶ **Proposition 3.4** (least fixpoint from min-decompositions). *Let Y be a finite set, $\mathbb{M}$ a complete MV-chain, $f : \mathbb{M}^Y \to \mathbb{M}^Y$ a monotone function and let $H_{\min} : Y \to \mathcal{P}_{fin}(\mathbb{M}^Y \to \mathbb{M})$ be a min-decomposition of f. Then $\mu f = \min\{\mu f_C \mid C \text{ is a strategy in } H_{\min}\}$.*

Although we do not focus on complexity issues, we observe that – under suitable assumptions – we can show that given a function f as a min-decomposition, the problem of checking whether $\mu f \sqsubseteq b$ for some bound $b \in \mathbb{M}^Y$ is in NP. For each $y \in Y$ we can nondeterministically guess $C(y) \in H_{\min}(y)$ thus defining a strategy. Assuming that the computation of μf_C is polynomial, we can thus determine in non-deterministic polynomial time (in the size of the representation of f) whether $\mu f \sqsubseteq \mu f_C \sqsubseteq b$.

Now in order to compute the least fixpoint, the idea is to start from some (arbitrary) strategy, say C_0, in $H_{\min}$. At each iteration, if the current strategy is C_i one tries to construct, on the basis of μf_{C_i}, a new strategy C_{i+1} which improves C_i, in the sense that $\mu f_{C_{i+1}}$ becomes smaller. This motivates the notion of improvement.

▶ **Definition 3.5** (min-improvement). *Let $f : \mathbb{M}^Y \to \mathbb{M}^Y$ be a monotone function, where Y is a finite set and $\mathbb{M}$ a complete MV-chain, and let $H_{\min}$ be a min-decomposition. Given strategies C, C' in $H_{\min}$, we say that C' is a* min-improvement *of C if $f_{C'}(\mu f_C) \sqsubset \mu f_C$. It is called a* stable *min-improvement if in addition $C'(y) = C(y)$ for all $y \in Y$ such that $f_{C'}(\mu f_C)(y) = \mu f_C(y)$. We denote by $imp_{\min}(C)$ (respectively $imp^s_{\min}(C)$) the set of (stable) min-improvements of C.*

The notion of stability will turn out to be useful later, for performing strategy iteration from below (as explained in the next section). In a stable min-improvement, the player is only allowed to switch the strategy in a state if this yields a strictly better payoff. Interestingly,

instances of this notion are adopted, more or less implicitly, in other strategy improvement algorithms in the literature (cf. [1, Definition 13] and the way in which improvements are computed in [10]). Clearly $imp^s_{\min}(C) \subseteq imp_{\min}(C)$. In addition, it can be easily seen that there exists a stable min-improvement as long as there is any improvement.

▶ Remark 3.6 (obtaining min-improvements). For a strategy C, if $imp_{\min}(C) \neq \varnothing$, one can obtain a min-improvement of C by taking $C' \neq C$ defined as $C'(y) = \arg\min_{h \in H_{\min}(y)} h(\mu f_C)$ and a stable min-improvement as:

$$C'(y) = \begin{cases} C(y) & \text{if } f(\mu f_C)(y) = \mu f_C(y) \\ \arg\min_{h \in H_{\min}(y)} h(\mu f_C) & \text{otherwise} \end{cases}$$

There could be several $h \in H_{\min}(y)$ where $h(\mu f_C)$ is minimal. Any such choice is valid.

We next show that, as suggested by the terminology, a min-improvement leads to a smaller least fixpoint.

▶ **Lemma 3.7** (min-improvements reduce fixpoints). *Let Y be a finite set, $\mathbb{M}$ a complete MV-chain, $f : \mathbb{M}^Y \to \mathbb{M}^Y$ a monotone function and $H_{\min}$ a min-decomposition of f. Given a strategy C in $H_{\min}$ and a min-improvement $C' \in imp_{\min}(C)$ it holds $\mu f_{C'} \sqsubset \mu f_C$.*

Thus, once the strategy can be improved, we will get closer to the least fixpoint of f. We next show that an improvement of the current strategy exists as long as we have not encountered a fixpoint of f.

▶ **Lemma 3.8** (min-improvements exist for non-fixpoints). *Let Y be a finite set, $\mathbb{M}$ a complete MV-chain, $f : \mathbb{M}^Y \to \mathbb{M}^Y$ a monotone function and $H_{\min}$ a min-decomposition. Given a strategy C in $H_{\min}$, the following are equivalent:*

1. $\mu f_C \notin Fix(f)$
2. $imp_{\min}(C) \neq \varnothing$
3. $f(\mu f_C) \sqsubset \mu f_C$

The above result suggests an algorithm for computing a fixpoint of a function $f : \mathbb{M}^Y \to \mathbb{M}^Y$ on the basis of some min-decomposition. The idea is to guess some strategy C, determine μf_C and check $imp_{\min}(C)$. If this set is empty we have reached some fixpoint, otherwise choose $C' \in imp_{\min}(C)$ for the next iteration. Note that for this algorithm it is irrelevant whether we use min-improvements or restrict to stable min-improvements. We also note that this procedure and the developed theory to this point work for monotone functions $f \colon \mathbb{L}^Y \to \mathbb{L}^Y$ where $\mathbb{L}$ is a complete lattice.

When we are interested in the least fixpoint and the function admits many fixpoints, the sketched algorithm determines a fixpoint which might not be the desired one. Exploiting the theory from [5], summarised in §2, we can refine the algorithm to ensure that it computes μf. For this, we have to work with non-expansive functions $f : \mathbb{M}^Y \to \mathbb{M}^Y$ with $\mathbb{M}$ being a complete MV-chain and Y a finite set. In fact, in this setting, given a fixpoint of f, say $a \in \mathbb{M}^Y$, relying on Theorem 2.3, we can check whether it is the least fixpoint of f. In case it is not, we can "improve" it obtaining a smaller pre-fixpoint of f in a way that we can continue the iteration from there. The resulting algorithm is reported in Fig. 2. Observe that in step 2b we clearly do not need to compute all improvements. Rather, a min-improvement, whenever it exists, can be determined, on the basis of Definition 3.5, using μf_{C_i} computed in step 2a. Moreover step 2c relies on Theorem 2.3 and Lemma 2.4.

▶ **Theorem 3.9** (least fixpoint, from above). *Let Y be a finite set, $\mathbb{M}$ a complete MV-chain, $f : \mathbb{M}^Y \to \mathbb{M}^Y$ be a non-expansive function and let $H_{\min}$ be a min-decomposition of f. The algorithm in Fig. 2 terminates and computes μf.*

1. Initialize: guess a strategy C_0, $i := 0$
2. **iterate**
   a. determine μf_{C_i}
   b. **if** $imp_{\min}(C_i) \neq \varnothing$, let $C_{i+1} \in imp_{\min}(C_i)$; $i := i + 1$; **goto** (a)
   c. **else if** $\mu f_{C_i} \neq \mu f$ let $a \sqsubset \mu f_{C_i}$ be a pre-fixpoint of f and determine C_{i+1} via

$$C_{i+1}(y) = \arg \min_{h \in H_{\min}(y)} h(a)$$

   $i := i + 1$; **goto** 2.(a)
   d. **else stop**

Figure 2 Computing the least fixpoint, from above.

Termination easily follows from the fact that the number of strategies is finite (since Y is finite and $H_{\min}(y)$ is finite for all $y \in Y$). Given that at any iteration the fixpoint decreases, no strategy can be considered twice, and thus the number of iterations is bounded by the number of strategies.

▶ **Example 3.10.** Let us revisit Example 3.3 and the fixpoint function $\mathcal{V}$ defined there. Its least fixpoint satisfies $\mu\mathcal{V}(\mathbf{1}) = 1$ and $\mu\mathcal{V}(v) = \varepsilon$ for any $v \in V \backslash \{\mathbf{1}\}$.

The optimal strategy for Min is to choose $\boldsymbol{av}$ as its successor since this forces Max to exit the cycle formed by $\boldsymbol{min}, \boldsymbol{av}, \boldsymbol{max}$ to $\boldsymbol{\varepsilon}$, yielding a payoff of ε for these states. If Max would behave in a way that the play keeps cycling he would obtain a payoff of 0, which is suboptimal.

We now apply our algorithm. We start by guessing a strategy for Min, so we assume $C_0(\boldsymbol{min}) = h_{\mathbf{1}}$, i.e. $C_0 = C_1^{\min}$ (for the naming of the strategies we refer to Example 3.3). The least fixpoint $\mu\mathcal{V}_{C_0}$ can be found by solving the following linear program:

$$\min \sum_{v \in V} a(v) \qquad a(\mathbf{1}) = 1 \qquad a(\boldsymbol{\varepsilon}) = \varepsilon \qquad a(\boldsymbol{av}) = \frac{1}{2}a(\boldsymbol{min}) + \frac{1}{2}a(\boldsymbol{max})$$
$$a(\boldsymbol{max}) \geqslant a(\boldsymbol{\varepsilon}) \quad a(\boldsymbol{max}) \geqslant a(\boldsymbol{av}) \quad a(\boldsymbol{min}) = a(\mathbf{1})$$

with $0 \leqslant a(v) \leqslant 1$ for $v \in V$, which yields $\mu\mathcal{V}_{C_0}(\boldsymbol{\varepsilon}) = \varepsilon$ and $\mu\mathcal{V}_{C_0}(v) = 1$ for all $v \in V \backslash \{\boldsymbol{\varepsilon}\}$. Now $\mu\mathcal{V}_{C_0}$ is a fixpoint of $\mathcal{V}$ – but not the least – and thus we find the vicious cycle formed by $\boldsymbol{min}, \boldsymbol{av}, \boldsymbol{max}$, i.e. $\nu\mathcal{V}_{\#}^{\mu\mathcal{V}_{C_0}} = \{\boldsymbol{min}, \boldsymbol{av}, \boldsymbol{max}\}$ and decrease the values of those states in a by δ, i.e. we obtain $a = \mu\mathcal{V}_{C_0} \ominus \delta_{\{\boldsymbol{min}, \boldsymbol{av}, \boldsymbol{max}\}}$. This results in $a(\mathbf{1}) = 1$, $a(\boldsymbol{\varepsilon}) = \varepsilon$ and $a(v) = 1 - \delta$ for all $v \in V \backslash \{\mathbf{1}, \boldsymbol{\varepsilon}\}$. Any $\delta \in (0, 1 - \varepsilon]$ is a valid choice.

Computing $C_1(y) = \arg\min_{h \in H_{\min}(y)} h(a)$ yields the strategy $C_1 = C_2^{\min}$, i.e. $C_1(\boldsymbol{min}) = h_{\boldsymbol{av}}$. By linear programming (replace $a(\boldsymbol{min}) = a(\mathbf{1})$ by $a(\boldsymbol{min}) = a(\boldsymbol{av})$) we obtain $\nu\mathcal{V}_{\#}^{\mu f_{C_1}} = \varnothing$, thus $\mu\mathcal{V}_{C_1} = \mu\mathcal{V}$ and the algorithm terminates.

3.3 Strategy iteration from below

Here we present a different generalized strategy iteration algorithm approaching the least fixpoint from below. Intuitively, now it is player Max who improves his strategy step by step, creating an ascending chain of least fixpoints which reaches the least fixpoint of the underlying function f. Despite the fact that in this case we cannot get stuck at a fixpoint which is not the least, the correctness argument is more involved.

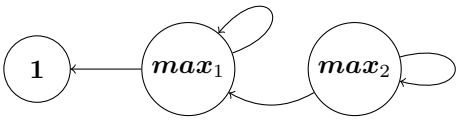


Figure 3 An example of a simple stochastic game where state **1** has payoff 1.

We will deal with max-decompositions of a function and we will need a notion of (stable) max-improvement which is naturally defined as a dualisation of the notion of (stable) min-improvement (Definition 3.5).

▶ **Definition 3.11** (max-improvement). *Let Y be a finite set, $\mathbb{M}$ a complete MV-chain, $f : \mathbb{M}^Y \to \mathbb{M}^Y$ a monotone function and $H_{\max}$ a max-decomposition. Given C, C' strategies in $H_{\max}$, we say that C' is a* max-improvement *of C if $\mu f_C \sqsubset f_{C'}(\mu f_C)$. It is called a* stable max-improvement *if in addition $C'(y) = C(y)$ for all $y \in Y$ such that $f_{C'}(\mu f_C)(y) = \mu f_C(y)$. We denote by $imp_{\max}(C)$ (respectively $imp^s_{\max}(C)$) the set of (stable) max-improvements of C.*

When iterating from above it was rather easy to show that given a strategy C and a min-improvement C', the latter yields a smaller least fixpoint $\mu f_{C'} \sqsubset \mu f_C$ (Lemma 3.7). Observing that μf_C is a pre-fixpoint of $f_{C'}$ was enough to prove this.

Here, however, we cannot simply dualise the argument. If C' is a max-improvement of C, we obtain that μf_C is a post-fixpoint of $f_{C'}$ which, in general, does not guarantee $\mu f_{C'} \sqsupset \mu f_C$. We have to resort to stable max-improvements and, in order to show that such improvements in fact yield greater least fixpoints, we need, again, to use the theory reviewed in §2. Hence, we have to work with non-expansive functions $f \colon \mathbb{M}^Y \to \mathbb{M}^Y$ where $\mathbb{M}$ is a complete MV-chain.

▶ **Lemma 3.12** (max-improvements increase fixpoints). *Let Y be a finite set, $\mathbb{M}$ a complete MV-chain, $f : \mathbb{M}^Y \to \mathbb{M}^Y$ a non-expansive function and $H_{\max}$ a max-decomposition. Given a strategy C in $H_{\max}$ and a stable max-improvement $C' \in imp^s_{\max}(C)$, then $\mu f_C \sqsubset \mu f_{C'}$.*

▶ **Example 3.13.** We note that working with max-improvements which are stable is essential for the validity of Lemma 3.12 above. In fact, consider the SSG in Figure 3 where $\boldsymbol{max}_1, \boldsymbol{max}_2 \in MAX$ and $\mathbf{1} \in SINK$, with reward 1. Let C be the strategy for Max where $\boldsymbol{max}_1$ and $\boldsymbol{max}_2$ have as successors $\mathbf{1}$ and $\boldsymbol{max}_2$, respectively. It is easy to see that $\mu\mathcal{V}_C(\mathbf{1}) = \mu\mathcal{V}_C(\boldsymbol{max}_1) = 1$ and $\mu\mathcal{V}_C(\boldsymbol{max}_2) = 0$. Now, an improvement in $imp_{\max}(C)$ can be the strategy C' which chooses $\boldsymbol{max}_1$ as a successor for both $\boldsymbol{max}_1$ and $\boldsymbol{max}_2$. Then we have $\mu\mathcal{V}_{C'}(\boldsymbol{max}_1) = \mu\mathcal{V}_{C'}(\boldsymbol{max}_2) = 0$, hence $\mu\mathcal{V}_C \sqsupset \mu\mathcal{V}_{C'}$. The reason why this happens is that C' is not a stable improvement of C since it uselessly changes the successor of $\boldsymbol{max}_1$ from $\mathbf{1}$ to $\boldsymbol{max}_1$, both mapped to 1 by $\mu\mathcal{V}_C$. A stable improvement of C is C'' where $\boldsymbol{max}_1$ and $\boldsymbol{max}_2$ have as successors $\mathbf{1}$ and $\boldsymbol{max}_1$, respectively. Then it can be seen that $\mu\mathcal{V}_{C''}(v) = 1$ for all states.

Relying on Lemma 3.12, we can easily prove the dual of Lemma 3.8, showing that a strategy admits a stable max-improvement as long as we have not reached a fixpoint of f.

▶ **Lemma 3.14** (max-improvements exist for non-fixpoints). *Let Y be a finite set, $\mathbb{M}$ a complete MV-chain, $f : \mathbb{M}^Y \to \mathbb{M}^Y$ a monotone function and $H_{\max}$ a max-decomposition. Given a strategy C in $H_{\max}$, the following are equivalent:*

1. $\mu f_C \notin Fix(f)$
2. $imp^s_{\max}(C) \neq \varnothing$
3. $\mu f_C \sqsubset f(\mu f_C)$

1. Initialize: guess a strategy C_0, $i := 0$
2. **iterate**
   a. determine μf_{C_i}
   b. **if** $imp^s_{\max}(C_i) \neq \varnothing$, let $C_{i+1} \in imp^s_{\max}(C_i)$; $i := i + 1$; **goto** (a)
   c. else **stop**

Figure 4 Computing the least fixpoint, from below.

To summarise, given a strategy C with $\mu f_C \notin Fix(f)$ we can construct a strategy C' with $\mu f_C \sqsubset \mu f_{C'}$. This creates an ascending chain of least fixpoints and since there are only finitely many strategies we will at some point find an optimal strategy C^* with $\mu f_{C^*} = \mu f$.

▶ **Proposition 3.15** (least fixpoint from max-decomposition). *Let Y be a finite set, $\mathbb{M}$ a complete MV-chain, $f : \mathbb{M}^Y \to \mathbb{M}^Y$ a non-expansive function and let $H_{\max} : Y \to \mathcal{P}_{fin}(\mathbb{M}^Y \to \mathbb{M})$ be a max-decomposition of f. Then $\mu f = \max\{\mu f_C \mid C \text{ is a strategy in } H_{\max}\}$.*

The above results lead us to a generalised strategy iteration algorithm which approaches the least fixpoint from below.

▶ **Theorem 3.16** (least fixpoint, from below). *Let Y be a finite set, $\mathbb{M}$ a complete MV-chain, $f : \mathbb{M}^Y \to \mathbb{M}^Y$ be a non-expansive function and let $H_{\max}$ be a max-decomposition of f. The algorithm in Fig. 4 terminates and computes μf.*

The iteration from below may seem more appealing since it cannot get stuck at any fixpoint of f. However, it has to be noted that the computation of μf_C – for a chosen strategy C – may be more difficult than before, which is illustrated by the following example.

▶ **Example 3.17.** Let us apply the above algorithm to the SSG in Example 3.3. Recall that the least fixpoint is given by $\mu\mathcal{V}(\mathbf{1}) = 1$ and $\mu\mathcal{V}(v) = \varepsilon$ for all $v \in V\backslash\{\mathbf{1}\}$.

We start by guessing a strategy for Max, so we assume $C_0(\boldsymbol{max}) = h_{\boldsymbol{av}}$, i.e. $C_0 = C_2^{\boldsymbol{max}}$. With this choice of strategy, Min is able to keep the game going infinitely in the cycle formed by $\boldsymbol{min}, \boldsymbol{av}, \boldsymbol{max}$ and thus payoff 0 is obtained. Now $\mu\mathcal{V}_{C_0}$ is given by $\mu\mathcal{V}_{C_0}(\boldsymbol{\varepsilon}) = \varepsilon$, $\mu\mathcal{V}_{C_0}(\mathbf{1}) = 1$ and $\mu\mathcal{V}_{C_0}(v) = 0$ for all $v \in V\backslash\{\boldsymbol{\varepsilon}, \mathbf{1}\}$. We note that $\mu\mathcal{V}_{C_0}$ cannot immediately be computed via linear programming, but there is a way to modify the fixpoint equation to have a unique fixpoint and hence linear programming can be used again [5]. This is done by precomputing states from which Min can force a non-terminating play and assigning payoff value 0 to them. Next, Max updates his strategy and we obtain $C_1 = C_1^{\max}$. As above we can compute $\mu\mathcal{V}_{C_1}$ – which, this time, equals $\mu\mathcal{V}$ – via linear programming.

▶ Remark 3.18. Given μf (without the corresponding strategy) an interesting question is how one can derive optimal strategies for Min or Max. Note that each presented strategy iteration algorithm only produces an optimal strategy for one player, but not for the other.

It is rather easy to find an optimal strategy with respect to $H_{\min}$. We can simply compute $C^*(y) = \arg\min_{h \in H_{\min}(y)} h(\mu f)$ which yields some optimal strategy C^*, i.e. $\mu f_{C^*} = \mu f$. It is enough to choose some minimum, even if this is ambiguous and there are several choices, each of which produces an optimal strategy. The strategy C^* is optimal since μf is a pre-fixpoint of f_{C^*} and $\mu f = \mu f_{C^*}$ follows from Proposition 3.4.

On the other hand, given μf, we cannot easily obtain an optimal strategy in $H_{\max}$. We will discuss in §4.1 (Example 4.1) that defining $C^*(y) = \arg\max_{h \in H_{\max}(y)} h(\mu f)$ for an arbitrary h where the value is maximal does not work in general.

4 Application: energy games

In this section we examine energy games [16] and show how both strategy iterations in §3 can be applied to solve energy games and have the advantage of providing us not only with the value vector, but also with a strategy, which is interesting, in particular, for Player 1 (Max).

Energy games are two-player games on finite graphs where Player 0 (Min) wants to keep the game going forever, while Player 1 (Max) wants it to stop eventually. Each state belongs to one player where he chooses the successor and each traversed edge reduces or increases the energy level by some integer. The game stops when an edge is taken which reduces the energy level below 0. The main question is how much initial energy is needed, such that Player 0 can keep the game going forever. It is possible to require an initial energy level of infinity.

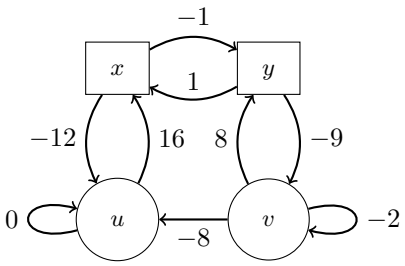
4.1 Introduction to energy games

A game graph is a tuple $\Gamma = (V_0, V_1, E, w)$ where $V = V_0 \cup V_1$, with $V_0 \cap V_1 = \varnothing$, is the set of states, $E \subseteq V \times V$ is the set of edges and $w\colon E \to \mathbb{Z}$ is a weight function. We assume that each state has at least one outgoing edge. We define $post(v) = \{v' \in V \mid (v, v') \in E\} \neq \varnothing$ for $v \in V$. States in V_0 and V_1 are owned by Player 0 and Player 1 respectively. Moving from state v to state v' will change the energy level by adding the value $w(v, v')$. If this value is positive, some energy is gained, otherwise energy decreases. It is the aim of Player 0 to keep the energy level from getting negative. An energy game is an infinite play on a game graph Γ and we note that optimal positional strategies exist for both players [16].

The solution of Γ is a function $g_\Gamma\colon V \to \mathbb{N}^\infty$ (where $\mathbb{N}^\infty = \mathbb{N} \cup \{\infty\}$) that assigns to each state the least energy level which is sufficient for Player 0 to keep the game going, independently of the chosen strategy of Player 1. It is known that the solution is the least fixpoint of the following function $\bar{\mathcal{E}}\colon (\mathbb{N}^\infty)^V \to (\mathbb{N}^\infty)^V$, defined as

$$\bar{\mathcal{E}}(a)(v) = \begin{cases} \min\limits_{v' \in post(v)} \max\{a(v') - w(v, v'), 0\} & \text{if } v \in V_0 \\ \max\limits_{v' \in post(v)} \max\{a(v') - w(v, v'), 0\} & \text{if } v \in V_1 \end{cases}$$

▶ **Example 4.1.** Consider the following energy game, where it is intended that circular and rectangular states belong to Player 0 and Player 1, respectively.



The optimal strategy for Player 0 is to choose u as the successor to u and v. Thus v requires an initial energy of 8 to keep going forever. For u an initial energy of 0 is sufficient. On the other hand, the optimal strategy for Player 1 is to choose y as successor to x and v as successor to y. This results in a required initial energy of 17 for y and 18 for x.

Thus, we obtain as least fixpoint $g_\Gamma(x) = 18$, $g_\Gamma(y) = 17$, $g_\Gamma(u) = 0$, $g_\Gamma(v) = 8$. Note that, if from u Player 0 would choose x, Player 1 could keep the game in a negative cycle.

Given only the least fixpoint g_Γ, the strategy of Player 1 is not deducible with a local reasoning, since from y the choices x, v are indistinguishable (in fact $g_\Gamma(y) = 17 = g_\Gamma(x) - 1 = g_\Gamma(v) - (-9)$). However, if x is chosen as successor to y (and still y as successor to x), we end up in a value vector where Min needs 0 initial energy in y to keep going.

4.2 Strategy iteration for energy games

In order to solve energy games in our framework, we have to consider non-expansive functions over MV-algebras, however $\mathbb{N}^\infty$ is unfortunately not an MV-algebra. For this, we use the results of [16], where it is shown that any energy game $\Gamma = (V_0, V_1, E, w)$ can be transformed into an energy game $\Gamma' = (V_0', V_1', E', w')$ with finite values only. Concretely, this is done by adding an "emergency exit" for each state in V_0 guaranteeing a finite amount of required energy to keep the game going forever. The solution $g_{\Gamma'}$ of Γ' satisfies $g_{\Gamma'}(v) < \infty$ for all $v \in V$ and the solution g_Γ of Γ can be easily reconstructed from $g_{\Gamma'}$. This allows us to restrict to energy games with finite values, where the solution is bounded by a suitable k. In this setting, letting $K = \{0, \dots, k\}$ and $\ominus_\mathbb{Z} \colon K \times \mathbb{Z} \to K$ given by $x \ominus_\mathbb{Z} y = \min\{\max\{x - y, 0\}, k\}$, we can define $\mathcal{E} \colon K^V \to K^V$ for $a \colon V \to K$ and $v \in V$ as

$$\mathcal{E}(a)(v) = \begin{cases} \min\limits_{(v,v') \in E} a(v') \ominus_\mathbb{Z} w(v, v') & \text{if } v \in V_0 \\ \max\limits_{(v,v') \in E} a(v') \ominus_\mathbb{Z} w(v, v') & \text{if } v \in V_1 \end{cases}$$

▶ **Lemma 4.2** (solution is least fixpoint of $\mathcal{E}$). *Let Γ be an energy game with finite values, bounded by k. Then $\mu\mathcal{E} = g_\Gamma$, i.e. the least fixpoint of $\mathcal{E}$ coincides with the solution of Γ.*

Recall from Example 2.1 that K is an MV-chain. Moreover $\mathcal{E} : K^V \to K^V$ can be proved to be non-expansive by showing that it can be expressed in terms of basic functions which are known or easily shown to be non-expansive and exploiting the fact that non-expansiveness is preserved by composition (see the full version [7]) and thus both generalised strategy iteration approaches in §3, from below and from above, can be applied for determining $\mu\mathcal{E}$, i.e., the solution of Γ.

Observe that the algorithms do not only compute $\mu\mathcal{E}$, but also provide an optimal strategy, for Player 0 when approaching from above and for Player 1 when approaching from below. The second case is of particular interest as it derives an optimal strategy for Player 1, which is often not treated in the literature (we are only aware of [10]).

We also remark that, when performing iteration from above or below, at each iteration, once a strategy C for Player 0 is fixed, we need to compute $\mu\mathcal{E}_C$. This can be done via linear programming, however it turns out that it is more efficient to use some form of value iteration, due to finiteness of the MV-algebra $\{0, \dots, k\}$.

The full version [7] spells out the approximation $\mathcal{E}^a_\#$ of the function $\mathcal{E}$ which – according to the theory in §2 – can be used for checking whether its least fixpoint has already been reached in strategy iteration from above. It also analyses known algorithms for solving energy games and compares their runtime to both kinds of strategy iteration. While other algorithms might in some cases have better runtimes, strategy iteration has the advantage of providing the optimal strategy.

5 Application: behavioural metrics for probabilistic automata

In this section we show how our technique can be used to compute behavioural distances over probabilistic automata. After introducing the necessary notions, we provide a min-decomposition of the corresponding function. The algorithm that we obtain by instantiating our generalised strategy iteration from above using such min-decomposition can be seen to be essentially the same as the one presented in [3].

▶ **Definition 5.1** (probabilistic automaton). *A* probabilistic automaton *(PA) is a tuple* $\mathcal{A} = (S, L, \delta, \ell)$ *consisting of a nonempty finite set* S *of states, a finite set of labels* L*, a successor function* $\delta : S \to \mathcal{P}_{fin}(\mathcal{D}(S))$ *and a labeling function* $\ell \colon S \to L$*.*

The idea is that from a state s, one can non-deterministically move to one of the probability distributions in $\delta(s)$.

The behavioural distance function is defined by combining Hausdorff and Kantorovich liftings for the nondeterministic and probabilistic parts, respectively. Recall that the *Kantorovich lifting* [30] $K : [0,1]^{Y\times Y} \to [0,1]^{\mathcal{D}(Y)\times\mathcal{D}(Y)}$ transforming a pseudometric d on Y to a pseudometric on $\mathcal{D}(Y)$ is defined, for $\beta, \beta' \in \mathcal{D}(Y)$, by

$$K(d)(\beta,\beta') = \min_{\omega\in\Omega(\beta,\beta')} \sum_{y,y'\in Y} d(y,y')\cdot\omega(y,y'),$$

where $\Omega(\beta,\beta')$ is the set of probabilistic couplings of β, β':

$$\Omega(\beta,\beta') = \{\omega \in \mathcal{D}(Y\times Y) \mid \forall y, y' \in Y \colon \sum_{x'\in Y} \omega(y,x') = \beta(y) \wedge \sum_{x\in Y} \omega(x,y') = \beta'(y')\}$$

Actually, the minimum is reached in one of the finitely many vertices of the polytope $\Omega(\beta,\beta')$, a set which we denote by $\Omega_V(\beta,\beta')$. The *Hausdorff lifting* $H : [0,1]^{Y\times Y} \to [0,1]^{\mathcal{P}(Y)\times\mathcal{P}(Y)}$ (in the variant of [26]) is defined, for $X, X' \in \mathcal{P}(Y)$, by

$$H(d)(X,X') = \min_{R\in\mathcal{R}(X,X')} \max_{(x,x')\in R} d(x,x'),$$

with $\mathcal{R}(X,X') = \{R \in \mathcal{P}(Y\times Y) \mid \pi_1(R) = X \wedge \pi_2(R) = X'\}$ the set-couplings of X, X' [26].

The rough idea is the following. If two states s and t have different labels they are at distance 1. Otherwise, in order to compute their distance one has find a "best match" between the outgoing transitions of such states, i.e., a set coupling as those considered in the Hausdorff lifting H. In turn, since, transitions are probabilistic, matching transitions means finding an optimal probabilistic coupling, as done by the Kantorovich lifting K, which is intuitively the best transport plan balancing the "supply" β and the "demand" β'. In this way the distance of s and t is expressed in terms of the distance of the states they can reach, hence, formally, behavioural distance is characterised as a least fixpoint.

▶ **Definition 5.2** (behavioural distance). *Let* $\mathcal{A} = (S, L, \delta, \ell)$ *be a PA. The behavioural distance on* $\mathcal{A}$ *is the least fixpoint of* $\mathcal{M} : [0,1]^{S\times S} \to [0,1]^{S\times S}$ *defined, for* $d \in [0,1]^{S\times S}$ *and* $s, t \in S$*, by* $\mathcal{M}(d)(s,t) = H(K(d))(\delta(s),\delta(t))$ *if* $\ell(s) = \ell(t)$ *and* $\mathcal{M}(d)(s,t) = 1$*, otherwise.*

▶ **Example 5.3.** Consider the probabilistic automaton in Fig. 5 with state space $Y = \{s,t,u\}$, labels $\ell(s) = \ell(t) = a$ and $\ell(u) = b$ and probability distributions $\beta_1, \beta_2, \beta_1', \beta_2', \beta''$ as indicated. For instance, from state s, there are two possible transitions β_1 which with probability $1/2$ goes to u and with probability $1/2$ stays in s, and β_2 which goes to t with probability 1.

In order to explain how function $\mathcal{M}$, resulting from the combination of Hausdorff and Kantorovich lifting, works, let us consider the pseudometric $d(s,t) = 1/2$, $d(s,u) = d(t,u) = 1$. This is not the least fixpoint, since the distance of states s, t is clearly 0 as the two states exihibit the same behaviour.

We now illustrate how to compute $\mathcal{M}(d)(s,t)$. We obtain $\mathcal{M}(d)(s,u) = \mathcal{M}(d)(t,u) = 1$ and, since $\ell(s) = \ell(t) = a$, we have

$$\mathcal{M}(d)(s,t) = H(K(d))(\delta(s),\delta(t)).$$

where $\delta(s) = \{\beta_1, \beta_2\}$ and $\delta(t) = \{\beta_1', \beta_2'\}$.

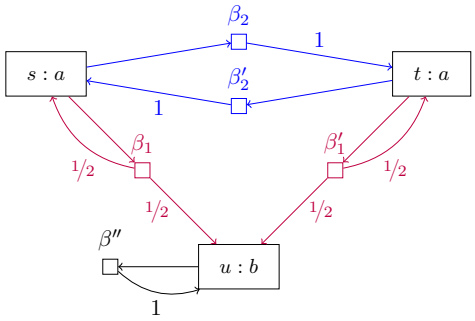


Figure 5 A probabilistic automaton.

It is relatively straightforward to see that the vertices of the coupling polytope $\Omega(\beta_1, \beta_1')$ are $\Omega_V(\beta_1, \beta_1') = \{\omega_1, \omega_2\}$ with

$$\omega_1(s,t) = 1/2,\ \omega_1(u,u) = 1/2 \quad \text{and} \quad \omega_2(s,u) = 1/2,\ \omega_2(u,t) = 1/2$$

and $\omega_i(x,y) = 0$, $i \in \{1,2\}$, for every other pair $(x,y) \in Y \times Y$. Then the Kantorovich lifting is determined as follows:

$$K(d)(\beta_1, \beta_1') = \min\{\sum_{x,y\in S} d(x,y)\cdot\omega_1(x,y), \sum_{x,y\in S} d(x,y)\cdot\omega_2(x,y)\} = \min\{1/4, 1\} = 1/4.$$

Similarly we can obtain $K(d)(\beta_1, \beta_2') = 1/2$, $K(d)(\beta_2, \beta_1') = 1/2$, $K(d)(\beta_2, \beta_2') = 1/4$.

In order to conclude the computation via the Hausdorff lifting, note that the minimal set-couplings of $\delta(s) = \{\beta_1, \beta_2\}$ and $\delta(t) = \{\beta_1', \beta_2'\}$ are

$$R_1 = \{(\beta_1, \beta_1'), (\beta_2, \beta_2')\} \qquad R_2 = \{(\beta_1, \beta_2'), (\beta_2, \beta_1')\}$$

and any other set-coupling includes R_1 or R_2. Then we obtain

$$\begin{aligned}
\mathcal{M}(d)(s,t) &= H(K(d))(\delta(s), \delta(t)) \\
&= \min\{\max_{(x,x')\in R_1} K(d)(x,x'), \max_{(x,x')\in R_2} K(d)(x,x')\} \\
&= \min\{\max\{K(d)(\beta_1,\beta_1'), K(d)(\beta_2,\beta_2')\}, \max\{K(d)(\beta_1,\beta_2'), K(d)(\beta_2,\beta_1')\}\} \\
&= \min\{\max\{1/4, 1/4\}, \max\{1/2, 1/2\}\} = \min\{1/4, 1/2\} = 1/4.
\end{aligned}$$

In order to cast this problem in our framework, we identify a suitable min-decomposition of $\mathcal{M}$. Observe that, for $d \in [0,1]^{S\times S}$ and $s,t \in S$ such that $\ell(s) = \ell(t)$, expanding the definitions of the liftings and taking advantage of complete distributivity, we have

$$\begin{aligned}
\mathcal{M}(d)(s,t) &= \min_{R\in\mathcal{R}(\delta(s),\delta(t))} \max_{(\beta,\beta')\in R} \min_{\omega\in\Omega_V(\beta,\beta')} \sum_{u,v\in S} d(u,v)\cdot\omega(u,v) \\
&= \min_{R\in\mathcal{R}(\delta(s),\delta(t))} \min_{f\in F_R} \max_{(\beta,\beta')\in R} \sum_{u,v\in S} d(u,v)\cdot f(\beta,\beta')(u,v)
\end{aligned}$$

where $F_R = \{f \colon R \to \mathcal{D}(S\times S) \mid f(\beta,\beta') \in \Omega_V(\beta,\beta') \text{ for } (\beta,\beta') \in R\}$, which is a finite set.

We can thus define a min-decomposition $H_{\min}$ for $\mathcal{M}$ (see Definition 3.1) such that $\mathcal{M}(d)(s,t) = \min_{h\in H_{\min}(s,t)} h(d)$ for all $s,t \in S$.

▶ **Definition 5.4** (min-decomposition of $\mathcal{M}$). *Let $\mathcal{A} = (S, L, \delta, \ell)$ be a PA. We denote by $H_{\min}$ the min-decomposition of $\mathcal{M}$ defined as follows. For $s, t \in S$ such that $\ell(s) = \ell(t)$, we let $H_{\min}(s,t) = \{h_{R,f} \mid R \in \mathcal{R}(\delta(s), \delta(t)), f \in F_R\}$, with $h_{R,f} : [0,1]^{S\times S} \to [0,1]$ defined as*

$$h_{R,f}(d) = \max_{(\beta,\beta')\in R} \sum_{u,v\in S} d(u,v) \cdot f(\beta,\beta')(u,v).$$

If instead $\ell(s) \neq \ell(t)$, we let $H_{\min}(s,t) = \{h_1\}$ where $h_1(d) = 1$ for all d.

A strategy C in $H_{\min}$ maps each pair of states $s, t \in S$ to a function in $H_{\min}(s,t)$, that is

- if $\ell(s) \neq \ell(t)$, to the unique element $h_1 \in H_{\min}(s,t)$;
- if $\ell(s) = \ell(t)$ to some $h_{R,f} \in H_{\min}(s,t)$, with $R \in \mathcal{R}(\delta(s), \delta(t))$ set-coupling and $f \in F_R$.

The decomposition above can be used to deduce that $\mathcal{M}$ is non-expansive and thus we can safely instantiate the algorithm in Fig. 2 to compute the least fixpoint from above. The resulting algorithm is quite similar to the one specifically developed for PAs in [3]. In particular, it can be seen that, apart from the different presentation, a strategy C corresponds to what [3] refers to as a *coupling structure*. In addition, the step in item (2c) of the algorithm (see Fig. 2) is analogous to that in [3]. In fact, in order to check whether the fixpoint obtained with the current strategy C_i, i.e. $\mu\mathcal{M}_{C_i}$, is the least fixpoint of $\mathcal{M}$, one considers the approximation $\mathcal{M}_{\#}^{\mu\mathcal{M}_{C_i}}$ and checks whether its greatest fixpoint is empty. Recalling that the post-fixpoints of $\mathcal{M}_{\#}^{\mu\mathcal{M}_{C_i}}$ have been shown in [5] to be the self-closed relations of [3], one derives that verifying the emptiness of the greatest fixpoint of $\mathcal{M}_{\#}^{\mu\mathcal{M}_{C_i}}$ corresponds exactly to checking whether the largest self-closed relation is empty (see [7] for more details).

6 Conclusion

We developed abstract algorithms for strategy iterations which allow to compute least fixpoints (or, dually, greatest fixpoints) of non-expansive functions over MV-algebras. The idea consists in expressing the function of interest as a minimum (or a maximum), and view the process of computing the function as a game between players Min and Max trying to minimise and maximise, respectively, the outcome. Then the algorithms proceed via a sequence of steps which converge to the least fixpoint from above, progressively improving the strategy of player Min, or from below, progressively improving the strategy of the player Max. The two procedures have similar worst-case complexity. The number of iterations is bounded by the number of strategies of the corresponding player $p \in \{\min, \max\}$, which is exponential in the input size (the number of strategies is $\prod_{y\in Y} |H_p(y)|$). This suggests that, depending on the setting, the fastest algorithm is the one using the smaller decomposition $H_{\min}$ respectively $H_{\max}$. However, a deeper analysis is still needed, as a smaller decomposition usually leads to a higher cost for computing μf_C.

The algorithms generalise an approach which has been recently proposed for simple stochastic games in [5, 6]. We showed how our technique instantiates to energy games, thus giving a method for determining the optimal strategies of both players, and to the computation of the behavioural distance for probabilistic automata, resulting in an algorithm similar to the non-trivial procedure in [3], which was also a source of inspiration.

Strategy iteration is used in many different application domains with fairly similar underlying ideas and we believe that it is fruitful to provide a general definition of the technique, clarifying and solving several issues on this level, such as the need for stable improvements or ways to deal with non-unique fixpoints.

There is an extremely wide literature on strategy iteration, often also referred to as policy iteration or strategy improvement (for an overview see [19]). As mentioned in the introduction, after its use on nonterminating stochastic games [23], it has been applied to solve many kinds of games, including discounted mean-payoff games [32], parity games [31, 28] and simple stochastic games [13]. Several quasi-polynomial algorithms have been recently devised for parity games [11, 22, 24], while the existence of a polynomial algorithm is still an important open problem. This has been generalized to finite lattices by [20].

Various papers on strategy iteration focus on lower bounds [18, 17]. Our paper, rather than concentrating on complexity issues, provides a general framework capturing strategy iteration in a general lattice theoretical setting. A work similar in spirit is [1] which proposes a meta-algorithm GSIA such that a number of strategy improvement algorithms for SSGs arise as instances, along with a general complexity bound. Differently from ours, this paper focuses on SSGs and iteration from below. However, it allows for the parametrisation of the algorithm on a subset of edges of interest in the game graph, which is not possible in our approach, and so it can provide interesting suggestions for further generalisations.

Another interesting setting of application is the lower-weak-upper-bound problem in mean-payoff games [9], reminiscent of energy games. For this problem, differently from the usual definition, the aim for one player is to maximise, never going negative, some resource which cannot exceed a given bound, while the other player has to minimise it. Also in this case, the solution can be computed as a least fixpoint. Due to the upper bound imposed to the resource, the function is not non-expansive, thus it is not captured by our theory. Still, the algorithm KASI proposed in [10], which computes the solution via strategy iteration, shares many similarities with our approach from below: at each iteration the algorithm computes a stable max-improvement of the current strategy. Indeed, when applying KASI to the special case where there is no upper bound to the accumulated resource, called lower-bound problem in [9] (also studied under different names in [12, 25]), the algorithm comes out as an exact instantiation of our general strategy iteration from below.

Given their generality, we believe that the algorithms proposed in the present paper have the potential to be applicable to a variety of other settings. In particular, some preliminary investigations show their applicability to computing behavioural metrics in an abstract coalgebraic setting [4]. Here the behavioural distance is naturally characterised as a least fixpoint of an operator based on the Wasserstein lifting of the behavioural functor. Then the idea is to view couplings used in the computation of the Wasserstein lifting as strategies and use strategy iteration for converging to the coalgebraic metric.

Our abstract strategy iteration algorithms rely on the assumption that, once a strategy for one of the players is fixed, the optimal "answering" strategy for the opponent can be computed efficiently. Identifying abstract settings where a min- or max-decompositions of a function ensures that the answering strategy can be indeed computed efficiently (e.g., via linear programming as it happens for simple stochastic games), is an interesting direction of future research.

References

1 David Auger, Xavier Badin de Montjoye, and Yann Strozecki. A generic strategy improvement method for simple stochastic games. In *Proc. of MFCS 2021*, volume 202 of *LIPIcs*, pages 12:1–12:22. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021.

2 Giorgio Bacci, Giovanni Bacci, Kim G. Larsen, and Radu Mardare. On-the-fly exact computation of bisimilarity distances. *Logical Methods in Computer Science*, 13(2:13):1–25, 2017.

3 Giorgio Bacci, Giovanni Bacci, Kim G. Larsen, Radu Mardare, Qiyi Tang, and Franck van Breugel. Computing probabilistic bisimilarity distances for probabilistic automata. *Logical Methods in Computer Science*, 17(1), 2021.

4 Paolo Baldan, Filippo Bonchi, Henning Kerstan, and Barbara König. Coalgebraic behavioral metrics. *Logical Methods in Computer Science*, 14(3), 2018. Selected Papers of the 6th Conference on Algebra and Coalgebra in Computer Science (CALCO 2015). URL: `https://lmcs.episciences.org/4827`.

5 Paolo Baldan, Richard Eggert, Barbara König, and Tommaso Padoan. Fixpoint theory – upside down. In *Proc. of FOSSACS 2021*, volume 12650 of *Lecture Notes in Computer Science*, pages 62–81. Springer, 2021.

6 Paolo Baldan, Richard Eggert, Barbara König, and Tommaso Padoan. Fixpoint theory – upside down. *CoRR*, abs/2101.08184, 2021. `arXiv:2101.08184`.

7 Paolo Baldan, Richard Eggert, Barbara König, and Tommaso Padoan. A lattice-theoretical view of strategy iteration, 2022. `arXiv:2207.09872`.

8 Filippo Bonchi, Barbara König, and Daniela Petrişan. Up-to techniques for behavioural metrics via fibrations. In *Proc. of CONCUR 2018*, volume 118 of *LIPIcs*, pages 17:1–17:17. Schloss Dagstuhl – Leibniz Center for Informatics, 2018. `doi:10.4230/LIPIcs.CONCUR.2018.17`.

9 Patricia Bouyer, Uli Fahrenberg, Kim G. Larsen, Nicolas Markey, and Jiří Srba. Infinite runs in weighted timed automata with energy constraints. In Franck Cassez and Claude Jard, editors, *Formal Modeling and Analysis of Timed Systems*, pages 33–47. Springer, 2008.

10 Luboš Brim and Jakub Chaloupka. Using strategy improvement to stay alive. *Electronic Proceedings in Theoretical Computer Science*, 25:40–54, 2010. `doi:10.4204/eptcs.25.8`.

11 Cristian S. Calude, Sanjay Jain, Bakhadyr Khoussainov, Wei Li, and Frank Stephan. Deciding parity games in quasipolynomial time. In *Proc. of STOC 2017*, pages 252–263, 2017.

12 Arindam Chakrabarti, Luca de Alfaro, Thomas A. Henzinger, and Mariëlle Stoelinga. Resource interfaces. In Rajeev Alur and Insup Lee, editors, *Prof. of EMSOFT '03 (International Workshop on Embedded Software)*, volume 2855 of *Lecture Notes in Computer Science*, pages 117–133. Springer, 2003.

13 Anne Condon. The complexity of stochastic games. *Information and Computation*, 96(2):203–224, 1992. `doi:10.1016/0890-5401(92)90048-K`.

14 Patrick Cousot and Radhia Cousot. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Proc. of POPL 1977*, pages 238–252. ACM, 1977.

15 Patrick Cousot and Radhia Cousot. Temporal abstract interpretation. In Mark N. Wegman and Thomas W. Reps, editors, *Proc. of POPL 2000*, pages 12–25. ACM, 2000.

16 Dani Dorfman, Haim Kaplan, and Uri Zwick. A faster deterministic exponential time algorithm for energy games and mean payoff games. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *Proc. of ICALP 2019*, volume 132 of *LIPIcs*, pages 114:1–114:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. `doi:10.4230/LIPIcs.ICALP.2019.114`.

17 John Fearnley. Exponential lower bounds for policy iteration. In *Proc. of ICALP 2010*, volume 6199 of *Lecture Notes in Computer Science*, pages 551–562. Springer, 2010.

18 Oliver Friedmann. An exponential lower bound for the latest deterministic strategy iteration algorithms. *Log. Methods Comput. Sci.*, 7(3), 2011.

19 Erich Grädel, Wolfgang Thomas, and Thomas Wilke, editors. *Automata, Logics, and Infinite Games: A Guide to Current Research*, volume 2500 of *Lecture Notes in Computer Science*. Springer, 2002.

20 Daniel Hausmann and Lutz Schröder. Quasipolynomial computation of nested fixpoints. In Jan Friso Groote and Kim Guldstrand Larsen, editors, *Proc. of TACAS 2021*, volume 12651 of *Lecture Notes in Computer Science*, pages 38–56. Springer, 2021.

21 Ronald A. Howard. *Dynamic Programming and Markov Processes*. MIT Press, 1960.

22 Marcin Jurdzinski and Ranko Lazic. Succinct progress measures for solving parity games. In *Proc. of LICS 2017*, pages 1–9. ACM/IEEE, 2017.

23 Richard M. Karp and Alan J. Hoffman. On nonterminating stochastic games. *Management Science*, 12(5):359–370, 1966.

24 Karoliina Lehtinen. A modal μ perspective on solving parity games in quasi-polynomial time. In *Proc. of LICS 2018*, pages 639–648. ACM/IEEE, 2018.

25 Yu. M. Lifshits and D. S. Pavlov. Potential theory for mean payoff games. *Zapiski Nauchnykh Seminarov POMI*, 340:61–75, 2006.

26 Facundo Mémoli. Gromov-Wasserstein distances and the metric approach to object matching. *Foundations of Computational Mathematics*, 11(4):417–487, 2011.

27 Daniele Mundici. MV-algebras. A short tutorial. Available at `http://www.matematica.uns.edu.ar/IXCongresoMonteiro/Comunicaciones/Mundici_tutorial.pdf`.

28 Sven Schewe. An optimal strategy improvement algorithm for solving parity and payoff games. In *Proc. of CSL 2008*, volume 5213 of *Lecture Notes in Computer Science*, pages 369–384. Springer, 2008.

29 Alfred Tarski. A lattice-theoretical theorem and its applications. *Pacific Journal of Mathematics*, 5:285–309, 1955.

30 Cédric Villani. *Optimal Transport – Old and New*, volume 338 of *A Series of Comprehensive Studies in Mathematics*. Springer, 2009.

31 Jens Vöge and Marcin Jurdzinski. A discrete strategy improvement algorithm for solving parity games. In *Proc. of CAV 2000*, volume 1855 of *Lecture Notes in Computer Science*, pages 202–215. Springer, 2000.

32 Uri Zwick and Mike Paterson. The complexity of mean payoff games on graphs. *Theoretical Computer Science*, 158(1&2):343–359, 1996.