Value Iteration*

Krishnendu Chatterjee[1] and Thomas A. Henzinger[1,2]

[1] University of California, Berkeley
[2] EPFL, Switzerland

Abstract. We survey value iteration algorithms on graphs. Such algorithms can be used for determining the existence of certain paths (*model checking*), the existence of certain strategies (*game solving*), and the probabilities of certain events (*performance analysis*). We classify the algorithms according to the value domain (boolean, probabilistic, or quantitative); according to the graph structure (nondeterministic, probabilistic, or multi-player); according to the desired property of paths (Borel level 1, 2, or 3); and according to the alternation depth and convergence rate of fixpoint computations.

1 Introduction

Symbolic model checking is an instance of value iteration on graphs. In value iteration, each vertex of a graph is assigned a value, and the values are iteratively improved until a fixpoint is reached. The improvement function is local, meaning that the new, improved value at a vertex depends on the old values at neighboring vertices. For symbolic model checking, the value domain is a boolean algebra of atomic propositions. Termination is guaranteed if the number of vertices is finite.

We take a systematic look at value iteration along four dimensions. First, we consider three different *value domains*. In the boolean domain, a value represents the truth or falsehood of atomic propositions. In the probabilistic domain, a value represents a probability. In the quantitative domain, a value is a real number (possibly greater than 1) which represents a reward, i.e., some quantitative information associated with a vertex. In the two nonboolean cases, the termination of value iteration is not guaranteed even for finite sets of vertices. However, an acceleration towards the fixpoint may be possible to ensure finite convergence. If even this proves difficult, then we investigate whether by value iteration the fixpoint can be approximated within any given error bound.

Second, as carrier of values, we consider three different kinds of *graph structures*, and their combinations. Simple graphs are nondeterministic generators of paths. Game graphs generate paths according to the competitive decisions made by two players. Probabilistic graphs represent stochastic processes that generate paths. Over simple graphs, value iteration can be used to determine the existence of certain paths, e.g., paths that satisfy a specification; over game graphs,

* This research was supported in part by the Swiss National Science Foundation and by the NSF grants CCR-0225610 and CCR-0234690.

it can be used to determine the existence of certain strategies, e.g., strategies that achieve an objective; and over probabilistic graphs, it can be used to determine the probability of certain events, where an event is a measurable set of paths.

Third, we consider increasingly complex specifications, objectives, and events; for uniformity, we refer to all of these as *objectives*. Objectives can be classified according to their Borel complexity. The values of Borel level-1 objectives are determined by finite paths. For example, reachability and safety specifications are Borel level-1, and so are the objectives to maximize or minimize a one-time reward. The Borel level-2 objectives are the simplest objectives whose values depend on infinite paths. For example, deterministic Büchi and coBüchi specifications are Borel level-2, and so are the objectives to maximize or minimize an infinitely recurring reward (i.e., limsup and liminf objectives). More general kinds of objectives include ω-regular specifications (Borel level-$2\frac{1}{2}$), and objectives to maximize or minimize the average of infinitely many rewards (Borel level-3).

Fourth, we consider three increasingly complex *value iteration* (or *fixpoint computation*) *schemes*. Alternation-free value iteration computes an increasing or decreasing sequence of values at each vertex. Value iteration of alternation depth-1 computes an increasing sequence of decreasing value sequences (or vice versa). General value iteration arbitrarily alternates the successive approximation of least and greatest fixpoints. Alternation-free value iteration can be used to compute the values of Borel level-1 objectives (e.g., symbolic CTL model checking; solving reachability and safety games). Value iteration of alternation depth-1 can be used to compute the values of Borel level-2 objectives (e.g., symbolic CTL model checking on structures with weak-fairness constraints; solving Büchi and coBüchi games). General value iteration can be used to compute the values of ω-regular objectives (e.g., symbolic CTL model checking on structures with strong fairness constraints; LTL model checking; solving parity games). However, by adjusting the value domain, even the values of complex Borel objectives (e.g., parity games and limit-average games) can be computed by an alternation-free value iteration. In this paper we only survey and generalize some known results; there remains much room for a detailed investigation of the connections and trade-offs between the complexity of the value domain, the Borel level of the objective, and the alternation depth of the fixpoint computation.

Section 2 defines the graph structures we consider. Sections 3 and 4 present alternation-free value iteration for Borel level-1 objectives, and alternation depth-1 value iteration for Borel level-2 objectives. Section 5 provides some remarks on more general objectives. Section 6 concludes with thoughts on related topics, such as strategy iteration (as opposed to value iteration) and discounting.

2 Graph Models of Systems

The states and transitions of a system can be viewed as vertices and edges of a graph. Often the states carry values, such as truth values for observable propositions, or quantitative values that represent resource data (e.g., buffer size, power consumption). This leads us to the model of valued graphs.

2.1 Valued graphs

A *valued graph* (S, E, D) consists of the following components.

1. A finite set S of *states*.
2. A binary *transition relation* $E \subseteq S \times S$. For a state $s \in S$, we write $E(s) = \{s' \in S \mid (s, s') \in E\}$ for the set of *successors*. We require that every state has at least one successor; that is, $E(s) \neq \emptyset$ for all $s \in S$.
3. A complete lattice D of *values*. In the cases that we consider in this paper, the value set D is a subset of the real numbers, and the lattice order is the usual ordering $\leq$ on the reals. We will encounter the following three cases.
   Boolean The value set D is the set $\mathbb{B} = \{0, 1\}$ of booleans. The least upper bound is disjunction; the greatest lower bound, conjunction.
   Probabilistic The value set D is the closed interval $[0, 1]$ of reals between 0 and 1. The least upper bound is max (for infinite sets, sup); the greatest lower bound, min (for infinite sets, inf).
   Quantitative The value set D is the set $\mathbb{R}^{\infty}_{\geq 0} = \mathbb{R}_{\geq 0} \cup \{\infty\}$ of nonnegative reals together with the top element ∞. Upper and lower bounds are as in the probabilistic case.

Throughout the paper, we use $\boldsymbol{n} = |S|$ for the number of states, and $\boldsymbol{m} = |E|$ for the number of transitions. Note that $\boldsymbol{m} \geq \boldsymbol{n}$, because every state has a successor.

Valuations. A *valuation* is a function $v \colon S \to D$ that maps every state to a value. In the boolean case, where $D = \mathbb{B}$, a valuation corresponds to a set $v \subseteq S$ of states; in this case, for a valuation v and a state s, we use the two expressions "$v(s) = 1$" and "$s \in v$" interchangeably. We write V for the set of valuations. The ordering on values is lifted to valuations in a pointwise fashion: for two valuations $v_1, v_2 \in V$, we write $v_1 \leq v_2$ iff $v_1(s) \leq v_2(s)$ for all states $s \in S$. The valuations with ordering $\leq$ form a complete lattice. In this lattice, a *chain* is an infinite sequence $C = \langle v_0, v_1, v_2, \ldots \rangle$ of valuations such that either $v_0 \leq v_1 \leq v_2 \leq \cdots$, or $v_0 \geq v_1 \geq v_2 \geq \cdots$. In the former case, the chain is *increasing*, and $\lim C = \text{lub } C$ denotes its least upper bound; in the latter case, the chain is *decreasing*, and $\lim C = \text{glb } C$ denotes its greatest lower bound.

Objectives. A *path* is an infinite sequence $\langle s_0, s_1, s_2, \ldots \rangle$ of states such that $(s_i, s_{i+1}) \in E$ for all $i \geq 0$. We write Ω the set of paths, and Ω_s for the set of paths that start from a given state $s \in S$. An *objective* is a Borel function $W \colon \Omega \to D$ that maps every path to a value.[1] In the boolean case, an objective corresponds to a Borel set $W \subseteq \Omega$ of paths; in this case, for an objective W and a path ω, we use the two expressions "$W(\omega) = 1$" and "$\omega \in W$" interchangeably.

[1] We require objectives to be Borel (with respect to the Cantor topology on paths and the order topology on values) for measurability.

2.2 Generalized graphs

We define several extensions of valued graphs: deterministic games; probabilistic graphs; probabilistic games; and concurrent games.

Deterministic games. A *deterministic game* consists of (1) a valued graph (S, E, D) and (2) a partition (S_1, S_2) of the set S of states into two subsets, S_1 and S_2. We refer to the states in S_1 as *player-1 states*; and to the states in S_2, as *player-2 states*. At player-1 states, player 1 chooses a successor; at player-2 states, player 2 chooses a successor.

Probabilistic graphs. A *probabilistic graph* consists of (1) a valued graph (S, E, D); (2) a partition (S_1, S_*) of the set S of states into the two subsets S_1 (player-1 states) and S_* (*probabilistic states*); and (3) a probabilistic transition function δ: $S_* \to \mathrm{Dist}(S)$ that maps every probabilistic state to a probability distribution of successors.[2] We require that for all states $s \in S_*$ and $s' \in S$, we have $(s, s') \in E$ iff $\delta(s)(s') > 0$. At a probabilistic state s, a successor $s' \in E(s)$ is chosen with probability $\delta(s)(s')$. The probabilistic graphs are commonly known as *Markov decision processes*.

Probabilistic games. A *probabilistic game* consists of (1) a valued graph (S, E, D); (2) a partition (S_1, S_2, S_*) of the set S of states into three subsets (player-1, player-2, and probabilistic states); and (3) a probabilistic transition function δ: $S_* \to \mathrm{Dist}(S)$. As for probabilistic graphs, we require that for all states $s \in S_*$ and $s' \in S$, we have $(s, s') \in E$ iff $\delta(s)(s') > 0$. Note that the deterministic games ($S_* = \emptyset$), the probabilistic graphs ($S_2 = \emptyset$), and the valued graphs (both $S_2 = \emptyset$ and $S_* = \emptyset$) are special cases of probabilistic games.

Concurrent games. The most general class of graph models we consider are the concurrent games. A *concurrent game* consists of (1) a valued graph (S, E, D); (2) two finite sets A_1 and A_2 of *player-1* and *player-2 moves*; and (3) a probabilistic transition function δ: $S \times A_1 \times A_2 \to \mathrm{Dist}(S)$. We require that for all states $s, s' \in S$, we have $(s, s') \in E$ iff $\delta(s, a_1, a_2)(s') > 0$ for some moves $a_1 \in A_1$ and $a_2 \in A_2$. Given a state $s \in S$ and two moves $a_1 \in A_1$ and $a_2 \in A_2$, let $E(s, a_1, a_2) = \{s' \in S \mid \delta(s, a_1, a_2)(s') > 0\}$. At a state s, both players choose moves simultaneously and independently; if player 1 chooses move $a_1 \in A_1$, and player 2 chooses $a_2 \in A_2$, then a successor $s' \in E(s, a_1, a_2)$ is chosen with probability $\delta(s, a_1, a_2)(s')$. The probabilistic games are equivalent to the special case of concurrent games where for all states $s \in S$ and all moves $a_1 \in A_1$ and $a_2 \in A_2$, either $\delta(s, a_1, a_2) = \delta(s, a_1, a_2')$ for all player-2 moves $a_2' \in A_2$, or $\delta(s, a_1, a_2) = \delta(s, a_1', a_2)$ for all player-1 moves $a_1' \in A_1$; that is, in each state only one of the two players can influence the choice of successor. This is why, in contrast to the more general concurrent games, the probabilistic games are also known as *turn-based games*.

[2] For a finite set X, we write $\mathrm{Dist}(X)$ for the set of probability distributions on X.

3 Level-1 Objectives and Alternation-free Value Iteration

The simplest kind of objectives are boolean reachability and safety objectives. In the quantitative case, these Borel level-1 objectives generalize to maximizing and minimizing objectives.

3.1 Maximizing and minimizing objectives

Consider a valued graph (S, E, D). A *reward function* $p: S \to D$ is a valuation; the value $p(s)$ at a state s is interpreted as a reward that is collected when s is visited. We assume that $p(s) > 0$ for some state $s \in S$. Given a reward function p, the *maximizing objective* $\mathsf{Max}(p): \Omega \to D$ is the function that maps every path to the maximal reward appearing along the path. Formally, for all paths $\omega = \langle s_0, s_1, s_2, \ldots \rangle$,

$$\mathsf{Max}(p)(\omega) \;=\; \max\{p(s_i) \mid i \geq 0\}.$$

In the boolean case, where $D = \mathbb{B}$, maximizing objectives are reachability objectives; they require a path to visit a *target* set p: we have $\omega \in \mathsf{Max}(p)$ iff $s_i \in p$ for some $i \geq 0$. The *minimizing objective* $\mathsf{Min}(p): \Omega \to D$ is defined dually, by

$$\mathsf{Min}(p)(\omega) = \min\{p(s_i) \mid i \geq 0\}.$$

Boolean minimizing objectives are safety objectives; they require a path to stay in a *safe* set p: we have $\omega \in \mathsf{Min}(p)$ iff $s_i \in p$ for all $i \geq 0$. While the boolean $\mathsf{Max}(p)$ objective corresponds to the formula $\Diamond p$ of linear temporal logic (a logic that is interpreted over paths), the boolean $\mathsf{Min}(p)$ objective corresponds to the formula $\Box p$. Both maximizing and minimizing objectives lie on level 1 of the Borel hierarchy.

3.2 Value improvement

We refer to alternation-free value iteration as *value improvement*. The value improvement algorithm operates on a valued graph $G = (S, E, D)$ using two functions: an improvement function and a limit function.

Improvement functions. An *improvement function* $\mathsf{Imp}: V \to V$ is a function on valuations which satisfies the following requirements.

Monotone For all valuations $v_1, v_2 \in V$, if $v_1 \leq v_2$, then $\mathsf{Imp}(v_1) \leq \mathsf{Imp}(v_2)$.

Continuous For every chain $C = \langle v_0, v_1, v_2, \ldots \rangle$ of valuations, the monotonicity of Imp ensures that $\mathsf{Imp}(C) = \langle \mathsf{Imp}(v_0), \mathsf{Imp}(v_1), \mathsf{Imp}(v_2), \ldots \rangle$ is a chain. We require that $\mathsf{Imp}(\lim C) = \lim \mathsf{Imp}(C)$.

Directed Either $v \leq \mathsf{Imp}(v)$ for all valuations $v \in V$; or $v \geq \mathsf{Imp}(v)$ for all valuations $v \in V$. In the former case, the function Imp is *extensive*; in the latter case, *reductive*.

Algorithm 1 ValueImprovement

Input: valued graph G, improvement function Imp, limit function Lim, precision $\alpha \in \mathbb{R}_{\geq 0}$, and initial valuation $v^0 \in V$.
Output: valuation $v^* \in V$.

$i := 0$;
do {
$v^{i+1} := \mathsf{Imp}(v^i)$;
$i := i + 1$;
} until $\mathrm{diff}(v^{i-1}, v^i) \leq \alpha$;
return $v^* := \mathsf{Lim}(v^i, \alpha)$.

The improvement functions we consider satisfy also the property of *locality*, which is defined as follows: for all states $s \in S$ and all valuations $v_1, v_2 \in V$, if $v_1(s') = v_2(s')$ for all successors $s' \in E(s)$, then $\mathsf{Imp}(v_1)(s) = \mathsf{Imp}(v_2)(s)$. Locality states that the value of the improvement function at a state s only depends on the values of the states that are successors of s. Locality restricts the power of improvement functions.

Limit functions. We define a distance between valuations: for two valuations $v_1, v_2 \in V$, let $\mathrm{diff}(v_1, v_2) = \max\{|v_1(s) - v_2(s)| \mid s \in S\}$. A *limit function* $\mathsf{Lim}\colon V \times \mathbb{R}_{\geq 0} \to V$ maps each valuation $v \in V$ and real $\alpha \geq 0$ to a valuation $\mathsf{Lim}(v, \alpha)$ such that $\mathrm{diff}(\mathsf{Lim}(v, \alpha), v) \leq \alpha$; that is, at each state, the input and output values of Lim do not differ by more than α. In particular, if $\alpha = 0$, then $\mathsf{Lim}(v, \alpha) = v$.

The value improvement algorithm. The value improvement algorithm (Algorithm 1) takes as input a valued graph, an improvement function Imp, a limit function Lim, a precision $\alpha \in \mathbb{R}_{\geq 0}$, and an initial valuation $v^0 \in V$. Starting from the initial valuation, the algorithm iteratively "improves" the valuation by applying the directed improvement function Imp: it computes a prefix of the *improvement chain* $C(v^0, \mathsf{Imp}) = \langle v^0, v^1, v^2, \ldots \rangle$, where $v^{i+1} = \mathsf{Imp}(v^i)$ for all $i \geq 0$. For boolean values, the improvement chain converges in a finite number of steps —that is, $v^{i+1} = v^i$ for some $i \geq 0$— and (provided the precision α is less than 1) the algorithm returns $\mathsf{Lim}(v^{i+1}, \alpha) = v^i$, which is the limit of the improvement chain. However, for probabilistic and quantitative values, finite convergence is not guaranteed. This is where the precision and the limit function come into play. If $\mathrm{diff}(v^i, v^{i+1}) \leq \alpha$ for some $i \geq 0$, then the algorithm applies the limit function and returns the valuation $\mathsf{Lim}(v^{i+1}, \alpha)$. Thus, for precisions $\alpha > 0$, the algorithm may terminate even if the improvement chain does not converge in a finite number of steps.

Fixpoint characterization. Since $(V, \leq)$ is a complete lattice, and Imp is a monotone and continuous function, the limit $v^\infty = \lim C(v^0, \mathsf{Imp})$ of the improvement chain is a fixpoint of the improvement function, i.e., $\mathsf{Imp}(v^\infty) = v^\infty$.

We refer to v^∞ as the *improvement fixpoint*. By Kleene's fixpoint theorem, if Imp is extensive, then v^∞ is the least fixpoint of Imp above v^0; that is,

$$v^\infty \;=\; \mathrm{glb}\{v \in V \mid v \geq v^0 \text{ and } \mathsf{Imp}(v) = v\} \;=\; (\mu X \geq v^0)\ \mathsf{Imp}(X),$$

where the notation of the right-most expression is borrowed from the μ-calculus [10]. Symmetrically, if Imp is reductive, then v^∞ is the greatest fixpoint of Imp below v^0; that is,

$$v^\infty \;=\; \mathrm{lub}\{v \in V \mid v \leq v^0 \text{ and } \mathsf{Imp}(v) = v\} \;=\; (\nu X \leq v^0)\ \mathsf{Imp}(X).$$

Rate of convergence. In the limit, the improvement chain always converges to the improvement fixpoint. From an algorithmic perspective, the rate of convergence is important. Given a valued graph G, an improvement function Imp, a limit function Lim, and a complexity class $\mathbb{C}$, we are interested in the following three questions.

Finitely reachable fixpoint Does Algorithm 1 terminate for all initial valuations v^0 and precision $\alpha = 0$? Finite reachability asks if the improvement fixpoint is reached in finitely many iterations of the improvement function: for all $v^0 \in V$, does there exist an $i \geq 0$ such that $\mathsf{Imp}(v^i) = v^i$? If the answer is Yes, then $v^* = v^\infty$; that is, the algorithm returns the improvement fixpoint. We furthermore wish to know if the required number i of iterations, given as a function of the valued graph G and the initial valuation v^0, lies in the complexity class $\mathbb{C}$.

Finitely computable fixpoint Does there exist a precision $\alpha > 0$ such that for all initial valuations v^0, Algorithm 1 terminates and returns the improvement fixpoint $v^* = v^\infty$? Finite computability asks if the improvement fixpoint can be computed using the limit function after finitely many iterations of the improvement function: is there an $\alpha > 0$ such that for all $v^0 \in V$, we have (1) for all $i \geq 0$, if $\mathrm{diff}(v^i, v^{i+1}) \leq \alpha$, then $\mathsf{Lim}(v^{i+1}, \alpha) = v^\infty$; and (2) there exists an $i \geq 0$ such that $\mathrm{diff}(v^i, v^{i+1}) \leq \alpha$. If the answer is Yes, then when given a suitable $\alpha > 0$ as input, the algorithm returns the improvement fixpoint. More precisely we wish to know if there exists such a suitable α such that required number i of iterations, given as a function of the valued graph G and the initial valuation v^0, lies in the complexity class $\mathbb{C}$.

Finitely approximable fixpoint For every real $\varepsilon > 0$ and initial valuation $v^0 \in V$, there exists an $i \geq 0$ such that $\mathrm{diff}(v^i, v^\infty) \leq \varepsilon$. We wish to know if the required number i of iterations, given as a function of the valued graph G and the initial valuation v^0, lies in the complexity class $\mathbb{C}$. In other words, finite approximability asks if the improvement fixpoint can be approximated within error ε using only the resources (time or space) provided by the complexity class $\mathbb{C}$. If the answer is Yes, then Algorithm 1 can be run with precision $\alpha = 0$ and stopped after i iterations with output v^i, which is guaranteed to deviate from the improvement fixpoint v^∞ by at most ε.

Whenever the finite reachability of an improvement fixpoint is not ensured, we investigate its finite computability, i.e., the existence of a suitable limit function. In cases where finite computability cannot be guaranteed for any α, we study finite approximability. We will also address the finer algorithmic question whether a value improvement scheme can be implemented (possibly with auxiliary data structures) so that its running time matches the best known upper bound for computing (or ε-approximating) the improvement fixpoint.

3.3 Graphs

Graph values of objectives. On a valued graph $G = (S, E, D)$, every objective W defines a valuation $\mathsf{sup}\, W : S \to D$, namely,

$$\mathsf{sup}\, W(s) \;=\; \sup\{W(\omega) \mid \omega \in \Omega_s\}$$

for all states $s \in S$. We refer to $\mathsf{sup}\, W$ as the *graph valuation* of the objective W. The graph value of a maximizing objective $\mathsf{Max}(p)$ at a state s is the maximal reward that appears along any path from s. In the boolean case, for a state $s \in S$, we have $s \in \mathsf{sup}\,\mathsf{Max}(p)$ iff some path from s leads to a state in p; and $s \in \mathsf{sup}\,\mathsf{Min}(p)$ iff some path from s contains only states in p. In other words, the graph valuation of the boolean $\mathsf{Max}(p)$ objective corresponds to the formula $\exists\Diamond p$ of branching temporal logic (a logic that is interpreted over states); and the graph valuation of the boolean $\mathsf{Min}(p)$ objective corresponds to the formula $\exists\Box p$. The dual, universal interpretation of objectives can be defined by $\mathsf{inf}\, W(s) = \inf\{W(\omega) \mid \omega \in \Omega_s\}$; however, we will not further pursue this case, which is symmetric.

Maximizing and minimizing problems on graphs. Given a valued graph G and a reward function p, we wish to compute the graph valuations of the objectives $\mathsf{Max}(p)$ and $\mathsf{Min}(p)$ over G. In the boolean case, this corresponds to the model-checking problem for the branching temporal logic CTL [10].

Graph predecessor operator. The *graph predecessor operator* $\mathsf{maxPre}: V \to V$ is the function on valuations defined by

$$\mathsf{maxPre}(v)(s) \;=\; \max\{v(s') \mid s' \in E(s)\}$$

for all valuations $v \in V$ and all states $s \in S$; that is, the value of $\mathsf{maxPre}(v)$ at a state s is the maximal value of v at the states that are successors of s. In the boolean case, we have $s \in \mathsf{maxPre}(v)$ iff there exists a successor $s' \in E(s)$ such that $s' \in v$. The function maxPre is monotone, continuous, and local.

Graph valuations as improvement fixpoints. The graph valuations of maximizing and minimizing objectives can be computed as finitely reachable improvement fixpoints. Consider a reward function p, and the corresponding objectives $\mathsf{Max}(p)$ and $\mathsf{Min}(p)$. We define two improvement functions:

$$\begin{aligned} \mathsf{maxImp}(v)(s) &= \max\{v(s), \mathsf{maxPre}(v)(s)\}; \\ \mathsf{minImp}(v)(s) &= \min\{v(s), \mathsf{maxPre}(v)(s)\}; \end{aligned}$$

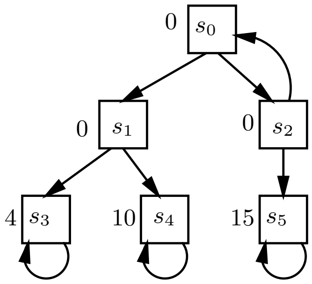

Fig. 1. Graph with maximizing objective.

for all valuations $v \in V$ and all states $s \in S$. Note that maxImp is extensive, and minImp reductive. In the boolean case, $\mathsf{maxImp}(v) = v \cup \mathsf{maxPre}(v)$ and $\mathsf{minImp}(v) = v \cap \mathsf{maxPre}(v)$. The improvement fixpoint $\lim C(p, \mathsf{maxImp})$ for the initial valuation p and the improvement function maxImp is the graph valuation $\sup \mathsf{Max}(p)$ of the maximizing objective. Similarly, the improvement fixpoint $\lim C(p, \mathsf{minImp})$ is the graph valuation $\sup \mathsf{Min}(p)$ of the minimizing objective. Both fixpoints are finitely reachable, and the number of improvement steps is bounded by the number of states. Hence the value improvement algorithm (Algorithm 1) with initial valuation $v^0 = p$, improvement function maxImp (resp. minImp), and precision $\alpha = 0$, returns the valuation $v^* = v^i = \sup \mathsf{Max}(p)$ (resp. $v^* = v^i = \sup \mathsf{Min}(p)$) after at most $i = \boldsymbol{n}$ iterations (regardless of the limit function, because $\alpha = 0$).

Example 1 (Graph with maximizing objective). Consider the valued graph shown in Figure 1. The reward function p is given by the state labels. We consider the objective $\mathsf{Max}(p)$ for player 1. The value improvement algorithm proceeds by computing value vectors, which consist of one value for each state. We denote by v^i the value vector at iteration i. The j-th component of v^i indicates the value for state s_{j-1} at iteration i. The initial value vector is given by p, that is, $v^0 = \langle 0, 0, 0, 4, 10, 15 \rangle$. Applying the improvement function maxImp to v^0, we obtain $v^1 = \langle 0, 10, 15, 4, 10, 15 \rangle$. This vector contains, for each state, the maximal reward that can be obtained in a single transition. Applying maxImp to v^1, we obtain $v^2 = \langle 15, 10, 15, 4, 10, 15 \rangle$, which indicates for each state the maximal reward that can be obtained in two transitions. Since $\mathsf{maxImp}(v^2) = v^2$, this is the improvement fixpoint; that is, the vector v^2 contains for each state the maximal reachable reward. $\square$

Optimality. Every improvement step (i.e., each application of the function maxImp or minImp) can be computed in $O(\boldsymbol{m})$ time. Hence a direct implementation of Algorithm 1 has the time complexity $O(\boldsymbol{mn})$. However, in the boolean case, the value improvement scheme can be implemented to run in $O(\boldsymbol{m})$ time. This is because each transition, once considered in the computation of some valuation v^i, is never revisited in subsequent iterations. Thus, in the boolean case, value improvement yields an optimal, linear-time algorithm for solving the maximizing and minimizing problems on graphs. In the quantitative case, the two problems can be solved in $O(\boldsymbol{m} + \boldsymbol{n} \log \boldsymbol{n})$ time by first sorting the rewards of all states, and then computing the states from which a reward can be reached, in descending (or ascending) order of rewards. We know of no implementation of value improvement which matches this time complexity.

3.4 Deterministic games

Deterministic strategies. Central to games is the notion of strategies. For games played on graphs, a strategy is a recipe that instructs a player which successor state to choose. Let $G = ((S, S_1, S_2), E, D)$ be a deterministic game. For player $k \in \{1, 2\}$, given a finite prefix of a path which represents the history of the game played so far, and which ends in a player-k state, a deterministic strategy for player k specifies how to extend the path by a transition. Formally, a *deterministic player-k strategy* is a function $\sigma\colon S^* \cdot S_k \to S$ such that $\sigma(w \cdot s) \in E(s)$ for all $w \in S^*$ and all $s \in S_k$. We write Σ^D and Π^D for the sets of deterministic player-1 and player-2 strategies, respectively. Given two deterministic strategies $\sigma \in \Sigma^D$ and $\pi \in \Pi^D$, and a state $s \in S$, we write $\omega_s^{\sigma,\pi}$ for the path $\langle s_0, s_1, s_2, \ldots \rangle \in \Omega$ such that (1) $s_0 = s$ and (2) for all $i \geq 0$, we have $\sigma(\langle s_0, s_1, \ldots, s_i \rangle) = s_{i+1}$ if $s_i \in S_1$, and $\pi(\langle s_0, s_1, \ldots, s_i \rangle) = s_{i+1}$ if $s_i \in S_2$; that is, if the game is started in state s and the two players play according to the strategies σ and π, then the result is the path $\omega_s^{\sigma,\pi}$.

Game values of objectives. On a deterministic game G, every objective W defines a valuation $\mathsf{supinf}\, W\colon S \to D$, namely,

$$\mathsf{supinf}\, W(s) \;=\; \sup_{\sigma \in \Sigma^D} \inf_{\pi \in \Pi^D} W(\omega_s^{\sigma,\pi}).$$

We refer to $\mathsf{supinf}\, W$ as the *deterministic game valuation* of the objective W. The game value of a maximizing objective $\mathsf{Max}(p)$ at a state s is the maximal reward that player 1 can ensure to appear along a path from s, against any strategy for player 2. In the boolean case, for a state $s \in S$, we have $s \in \mathsf{supinf}\, \mathsf{Max}(p)$ iff there exists a player-1 strategy σ such that for every player-2 strategy π, the path from s given σ and π leads to a state in p. Similarly, if $D = \mathbb{B}$, then $s \in \mathsf{supinf}\, \mathsf{Min}(p)$ iff there exists a player-1 strategy σ such that for every player-2 strategy π, the path from s given σ and π contains only states in p. Thus, the game valuation of the boolean $\mathsf{Max}(p)$ objective corresponds to the formula $\langle\!\langle 1 \rangle\!\rangle \Diamond p$ of the alternating temporal logic ATL [1]; and the game valuation of the boolean $\mathsf{Min}(p)$ objective corresponds to the ATL formula $\langle\!\langle 1 \rangle\!\rangle \Box p$.

Maximizing and minimizing problems on deterministic games. Given a deterministic game G and a reward function p, we wish to compute the deterministic game valuations of the objectives $\mathsf{Max}(p)$ and $\mathsf{Min}(p)$. In the boolean case, this corresponds to the model-checking problem for ATL.

Game predecessor operator. The *deterministic game predecessor operator* $\mathsf{maxminPre}\colon V \to V$ is the function on valuations defined by

$$\mathsf{maxminPre}(v)(s) \;=\; \begin{cases} \max\{v(s') \mid s' \in E(s)\} & \text{if } s \in S_1; \\ \min\{v(s') \mid s' \in E(s)\} & \text{if } s \in S_2; \end{cases}$$

that is, the value of $\mathsf{maxminPre}(v)$ at a player-1 state s is the maximal value of v at the successors of s, and at a player-2 state s it is the minimal value of v at the successors of s. In the boolean case, we have $s \in \mathsf{maxminPre}(v)$ iff (1) if $s \in S_1$, then there exists a successor $s' \in E(s)$ such that $s' \in v$; and (2) if $s \in S_2$, then $s' \in v$ for all successors $s' \in E(s)$. The function $\mathsf{maxminPre}$ is monotone, continuous, and local.

Deterministic game valuations as improvement fixpoints. As in the case of graphs, the deterministic game valuations of maximizing and minimizing objectives can be computed as finitely reachable improvement fixpoints. Consider a reward function p, and the corresponding objectives $\mathsf{Max}(p)$ and $\mathsf{Min}(p)$. We redefine the two improvement functions:

$$\begin{aligned} \mathsf{maxImp}(v)(s) &= \max\{v(s), \mathsf{maxminPre}(v)(s)\}; \\ \mathsf{minImp}(v)(s) &= \min\{v(s), \mathsf{maxminPre}(v)(s)\}. \end{aligned}$$

Note that maxImp is still extensive, and minImp reductive. The improvement fixpoint $\lim C(p, \mathsf{maxImp})$ for the initial valuation p and the improvement function maxImp is the deterministic game valuation $\mathsf{supinf}\,\mathsf{Max}(p)$ of the maximizing objective. Similarly, the improvement fixpoint $\lim C(p, \mathsf{minImp})$ is the deterministic game valuation $\mathsf{supinf}\,\mathsf{Min}(p)$ of the minimizing objective. Both fixpoints are finitely reachable, and as in the case of graphs, the number of improvement steps is bounded by the number of states. Hence the value improvement algorithm (Algorithm 1) with initial valuation $v^0 = p$, improvement function maxImp (resp. minImp), and precision $\alpha = 0$, returns the valuation $v^* = v^i = \mathsf{supinf}\,\mathsf{Max}(p)$ (resp. $v^* = v^i = \mathsf{supinf}\,\mathsf{Min}(p)$) after at most $i = \boldsymbol{n}$ iterations.

Example 2 (Deterministic game with maximizing objective). Consider the deterministic game shown in Figure 2. The $\Box$ states are player 1 states, and the $\Diamond$ state is a player 2 state (we will follow this graphical convention in all figures of this paper). We consider the objective $\mathsf{Max}(p)$ for player 1. From the initial valuation $v^0 = \langle 0, 0, 0, 4, 10, 15 \rangle$ given by the reward function p (indicated by the state labels), we obtain $v^1 = \langle 0, 10, 0, 4, 10, 15 \rangle$. Note that state s_2 chooses the successor with the minimum value, i.e., the value of s_0. Applying maxImp again, we obtain $v^2 = \langle 10, 10, 0, 4, 10, 15 \rangle$, and then $v^3 = \langle 10, 10, 10, 4, 10, 15 \rangle$. This is the improvement fixpoint, which represents the value of the deterministic maximizing game at each state. $\Box$

Optimality. The situation is similar to the case of graphs. Each application of maxImp or minImp can be computed in $O(\boldsymbol{m})$ time, yielding the time complexity

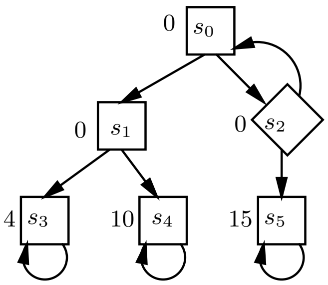

Fig. 2. Deterministic game with maximizing objective.

$O(\boldsymbol{mn})$ for Algorithm 1. In the boolean case, the value improvement scheme can be implemented to run in $O(\boldsymbol{m})$ time, which is optimal. Consider the case of a reachability (i.e., boolean maximizing) objective. We maintain a counter for each state s, which is initialized to 0 and incremented whenever a transition from s to some valuation v^i is visited. If $s \in S_1$, then s is included in v^{i+1} as soon as the counter becomes positive (i.e., some successor of s lies in v^i); if $s \in S_2$, then s is included in v^{i+1} when the counter reaches the outdegree of s (i.e., all successors of s lie in v^i). In this way, each transition is visited only once. In the quantitative case, the maximizing and minimizing problems on deterministic games can be solved in $O(\boldsymbol{m} + \boldsymbol{n} \log \boldsymbol{n})$ time [2], but we know of no implementation of value improvement which matches this bound.

3.5 Probabilistic graphs

Probabilistic strategies. A probabilistic strategy extends a finite path by a probability distribution of successors. Although probabilistic graphs do not contain player-2 states, we define probabilistic strategies right away for the more general case of probabilistic games. Let $((S, S_1, S_2, S_*), (E, \delta), D)$ be a probabilistic game, and let $k \in \{1, 2\}$. A *probabilistic player-k strategy* is a function $\sigma \colon S^* \cdot S_k \to \mathrm{Dist}(S)$ such that for all $w \in S^*$ and all $s, s' \in S_k$, if $\sigma(w \cdot s)(s') > 0$, then $s' \in E(s)$. We write Σ^P and Π^P for the sets of probabilistic player-1 and player-2 strategies, respectively. Given two probabilistic strategies $\sigma \in \Sigma^P$ and $\pi \in \Pi^P$, and a state $s \in S$, we write $\mathrm{Pr}_s^{\sigma,\pi}$ for the probability measure on the set Ω of paths which is defined inductively as follows: for all $w \in S^*$ and all $t, t' \in S$, (1) $\mathrm{Pr}_s^{\sigma,\pi}(t \cdot S^\omega)$ is 1 if $t = s$, and it is 0 otherwise; and (2) if $\mathrm{Pr}_s^{\sigma,\pi}(w \cdot t \cdot S^\omega) = x$, then $\mathrm{Pr}_s^{\sigma,\pi}(w \cdot t \cdot t' \cdot S^\omega)$ is $x \cdot \sigma(w \cdot t)(t')$ if $t \in S_1$, it is $x \cdot \pi(w \cdot t)(t')$ if $t \in S_2$, and it is $x \cdot \delta(t)(t')$ if $t \in S_*$. From this definition of $\mathrm{Pr}_s^{\sigma,\pi}$ for all basic open sets of paths, we obtain probabilities for all Borel sets. For an objective $W \subseteq \Omega$, we write $\mathrm{E}_s^{\sigma,\pi}[W]$ for the expected value of W under the probability measure $\mathrm{Pr}_s^{\sigma,\pi}$.

Example 3 (Probabilistic graph). Consider the probabilistic graph shown in Figure 3, with the single probabilistic state s_2 (probabilistic states are indicated by circles). The probabilities of the transitions from s_2 to s_0 and to s_5 are

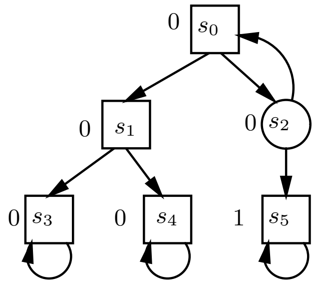

Fig. 3. Probabilistic graph with reachability objective.

both $\frac{1}{2}$. We consider the following three deterministic strategies σ_1, σ_2, and σ_3 for player 1: strategy σ_1 chooses at state s_0 the successor s_1, and at s_1 the successor s_3; strategy σ_2 chooses at s_0 the successor s_1, and at s_1 the successor s_4; and strategy σ_3 chooses at s_0 the successor s_2. Given strategy σ_1 and start state s_0, the outcome is the path $\omega_{s_0}^{\sigma_1} = s_0 \cdot s_1 \cdot s_3^{\omega}$; given σ_2 and s_0, the outcome is the path $\omega_{s_0}^{\sigma_2} = s_0 \cdot s_1 \cdot s_4^{\omega}$. Given strategy σ_3 and start state s_0, the outcome is a set of possible paths, namely, $((s_0 \cdot s_2)^* \cdot s_5^{\omega}) \cup (s_0 \cdot s_2)^{\omega}$. Observe that the probability of visiting s_2 for n times and never visiting s_5 is $(\frac{1}{2})^n$, which goes to 0 as n goes to ∞. Therefore $\Pr_{s_0}^{\sigma_3}((s_0 \cdot s_2)^* \cdot s_5^{\omega}) = 1$, and $\Pr_{s_0}^{\sigma_3}((s_0 \cdot s_2)^{\omega}) = 0$. $\square$

Probabilistic graph values of objectives. On a probabilistic graph $G = ((S, S_1, S_*), (E, \delta), D)$, an objective W defines the valuation $\mathsf{sup}\, W\colon S \to D$ given by

$$\mathsf{sup}\, W(s) \;=\; \sup\{\mathrm{E}_s^{\sigma}[W] \mid \sigma \in \Sigma^P\}.$$

We refer to $\mathsf{sup}\, W$ as the *probabilistic graph valuation* of the objective W. The probabilistic graph value of a maximizing objective $\mathsf{Max}(p)$ at a state s is the maximal expectation of the function $\mathsf{Max}(p)$ that player 1 can ensure from s. Note that if W is a boolean objective (that is, $W\colon \Omega \to \mathbb{B}$), in general the valuation $\mathsf{sup}\, W$ has probabilistic values (that is, $\mathsf{sup}\, W\colon S \to [0,1]$). If W is a quantitative objective (that is, $W\colon \Omega \to \mathbb{R}_{\geq 0}^{\infty}$), then $\mathsf{sup}\, W$ is a quantitative valuation.

Maximizing and minimizing problems on probabilistic graphs. Given a probabilistic graph G and a reward function p, we wish to compute the probabilistic graph valuations of the objectives $\mathsf{Max}(p)$ and $\mathsf{Min}(p)$.

Probabilistic graph predecessor operator. The *probabilistic graph predecessor operator* $\mathsf{maxPre}^P\colon V \to V$ is the function on valuations defined by

$$\mathsf{maxPre}^P(v)(s) \;=\; \begin{cases} \max\{v(s') \mid s' \in E(s)\} & \text{if } s \in S_1; \\ \sum_{s' \in E(s)} v(s') \cdot \delta(s)(s') & \text{if } s \in S_*; \end{cases}$$

that is, the value of $\mathsf{maxPre}^P(v)$ at a player-1 state s is the maximal value of v at the successors of s, and at a probabilistic state s it is the average value of v at the successors of s. Note that if v is a boolean valuation (that is, $v: S \to \mathbb{B}$), in general the predecessor valuation $\mathsf{maxPre}^P(v)$ has probabilistic values (that is, $\mathsf{maxPre}^P(v): S \to [0,1]$). If v is a quantitative valuation, then so is $\mathsf{maxPre}^P(v)$. In all cases, the function maxPre^P is monotone, continuous, and local.

Probabilistic graph valuations as improvement fixpoints. The probabilistic graph valuations of maximizing and minimizing objectives can be expressed as improvement fixpoints, but these fixpoints are not finitely reachable. Consider a reward function p, and the corresponding objectives $\mathsf{Max}(p)$ and $\mathsf{Min}(p)$. We redefine the two improvement functions:

$$\begin{aligned} \mathsf{maxImp}(v)(s) &= \max\{v(s), \mathsf{maxPre}^P(v)(s)\}; \\ \mathsf{minImp}(v)(s) &= \min\{v(s), \mathsf{maxPre}^P(v)(s)\}. \end{aligned}$$

Note that maxImp is still extensive, and minImp reductive. The improvement fixpoint $\lim C(p, \mathsf{maxImp})$ for the initial valuation $v^0 = p$ and the improvement function maxImp is the probabilistic graph valuation $\sup \mathsf{Max}(p)$ of the maximizing objective [16]. Similarly, the improvement fixpoint $\lim C(p, \mathsf{minImp})$ is the probabilistic graph valuation $\sup \mathsf{Min}(p)$ of the minimizing objective. Example 4 (below) shows that the two fixpoints are not finitely reachable.

Precision of values. We assume that all transition probabilities and rewards are given as rational numbers. From results of [12, 35] it follows that all values of the probabilistic graph valuations $\sup \mathsf{Max}(p)$ and $\sup \mathsf{Min}(p)$ are again rationals, and that the denominators can be bounded. Let $\delta_u = \max\{d \mid \delta(s)(s') = \frac{n}{d} \text{ for } s \in S_* \text{ and } s' \in E(s)\}$ be the largest denominator of all transition probabilities. Let $p_u = \mathrm{lcm}\{d \mid p(s) = \frac{n}{d} \text{ for } s \in S\}$ be the least common multiple of all reward denominators. Let $p_{\max} = \max\{n \mid p(s) = \frac{n}{d} \text{ for } s \in S\}$ be the largest numerator of all rewards; note that $p_{\max} > 0$ because at least one reward is positive. Then, for all states $s \in S$, both $\sup \mathsf{Max}(p)(s)$ and $\sup \mathsf{Min}(p)(s)$ have the form $\frac{n}{d}$ for a nonnegative integer n and a positive integer $d \leq \boldsymbol{\gamma}$, where

$$\boldsymbol{\gamma} = \delta_u^{4\boldsymbol{m}} \cdot p_u \cdot p_{\max}.$$

This *boundedness* property of probabilistic graph values for maximizing and minimizing objectives is the key for proving the finite computability of the two improvement fixpoints $\sup \mathsf{Max}(p) = \lim C(p, \mathsf{maxImp})$ and $\sup \mathsf{Min}(p) = \lim C(p, \mathsf{minImp})$. The value improvement algorithm (Algorithm 1) with initial valuation $v^0 = p$, improvement function maxImp (resp. minImp), and precision $\alpha = \frac{1}{2\boldsymbol{\gamma}}$, computes a valuation v^i such that $\mathrm{diff}(v^i, v^{i+1}) \leq \alpha$ within a number $i \in O(\boldsymbol{\gamma}^2)$ of iterations. In particular, the number of iterations depends exponentially on the number of states and on the size of rewards. The limit function Lim to obtain v^* from v^i is defined as follows: for each state $s \in S$, round the value $v^i(s)$ to the nearest multiple of $\frac{1}{\boldsymbol{\gamma}}$. Then v^* is the desired improvement fixpoint; that is, $v^* = \sup \mathsf{Max}(p)$ (resp. $v^* = \sup \mathsf{Min}(p)$).

Example 4 (Probabilistic graph with reachability objective). Recall the probabilistic graph from Figure 3, and consider the boolean maximizing objective to reach state s_5 (the reward at s_5 is 1, and the rewards at all other states are 0). We focus on the value improvements at the two states s_0 and s_2. Initially, both values are 0. After $2i$ improvement steps, both values are $\frac{2^i-1}{2^i}$. To see this, observe that in the $(2i+1)$-st iteration, the value at s_2 becomes

$$\frac{1}{2} \cdot \left(\frac{2^i - 1}{2^i} + 1 \right) \;=\; \frac{2^{i+1} - 1}{2^{i+1}};$$

and in the $(2i+2)$-nd iteration, the value at s_0 becomes $\frac{2^{i+1}-1}{2^{i+1}}$. Hence both values approach 1. Indeed, as we argued in Example 3, the player-1 strategy σ_3 ensures that the target state s_5 is reached with probability 1. Moreover, by the boundedness property of probabilistic graph values, once the values at s_0 and s_1 exceed $1 - \frac{1}{2^{37}}$, we can conclude that both values are 1. $\square$

Optimality. The maximizing and minimizing problems on probabilistic graphs can be solved in polynomial time using linear programming [18]. The linear-programming approach works equally for boolean and quantitative objectives. However, no better bound than an exponential bound in the number of states is known for the value improvement algorithm on probabilistic graphs, even in the special case of boolean reachability objectives.

3.6 Probabilistic games

Probabilistic game values of objectives. On a probabilistic game $G = ((S, S_1, S_2, S_*), (E, \delta), D)$, an objective W defines the valuation $\mathsf{supinf}\, W \colon S \to D$ given by

$$\mathsf{supinf}\, W(s) \;=\; \sup_{\sigma \in \Sigma^P} \inf_{\pi \in \Pi^P} \mathrm{E}_s^{\sigma,\pi}[W].$$

We refer to $\mathsf{supinf}\, W$ as the *probabilistic game valuation* of the objective W. The probabilistic game value of a maximizing objective $\mathsf{Max}(p)$ at a state s is the maximal expectation of the function $\mathsf{Max}(p)$ that player 1 can ensure from s against any strategy for player 2. The maximizing and minimizing problems on probabilistic games ask, given a probabilistic game G and a reward function p, to compute the probabilistic game valuations $\mathsf{supinf}\,\mathsf{Max}(p)$ and $\mathsf{supinf}\,\mathsf{Min}(p)$.

Probabilistic game predecessor operator. The *probabilistic game predecessor operator* $\mathsf{maxminPre}^P \colon V \to V$ is the function on valuations defined by

$$\mathsf{maxminPre}^P(v)(s) \;=\; \begin{cases} \max\{v(s') \mid s' \in E(s)\} & \text{if } s \in S_1; \\ \min\{v(s') \mid s' \in E(s)\} & \text{if } s \in S_2; \\ \sum_{s' \in E(s)} v(s') \cdot \delta(s)(s') & \text{if } s \in S_*. \end{cases}$$

The function $\mathsf{maxminPre}^P$ is monotone, continuous, and local.

Probabilistic game valuations as improvement fixpoints. Consider a reward function p, and the corresponding objectives $\mathsf{Max}(p)$ and $\mathsf{Min}(p)$. We redefine the two improvement functions:

$$\begin{aligned} \mathsf{maxImp}(v)(s) &= \max\{v(s), \mathsf{maxminPre}^P(v)(s)\}; \\ \mathsf{minImp}(v)(s) &= \min\{v(s), \mathsf{maxminPre}^P(v)(s)\}. \end{aligned}$$

Note that maxImp is still extensive, and minImp reductive. The improvement fixpoint $\lim C(p, \mathsf{maxImp})$ for the initial valuation $v^0 = p$ and the improvement function maxImp is the probabilistic game valuation $\mathsf{supinf}\,\mathsf{Max}(p)$ of the maximizing objective [16], and the improvement fixpoint $\lim C(p, \mathsf{minImp})$ is the probabilistic game valuation $\mathsf{supinf}\,\mathsf{Min}(p)$ of the minimizing objective. Since the probabilistic games generalize the probabilistic graphs, the two fixpoints are not finitely reachable. However, as in the case of probabilistic graphs, we have finite computability. If all transition properties and rewards are rational, then we can show the following boundedness property: for all states $s \in S$, both $\mathsf{sup}\,\mathsf{Max}(p)(s)$ and $\mathsf{sup}\,\mathsf{Min}(p)(s)$ have the form $\frac{n}{d}$ for a nonnegative integer n and a positive integer $d \leq \boldsymbol{\gamma}$, where the bound $\boldsymbol{\gamma}$ is defined as for probabilistic graphs [12, 35]. Furthermore, the value improvement algorithm (Algorithm 1) with initial valuation $v^0 = p$, improvement function maxImp (resp. minImp), and precision $\alpha = \frac{1}{2\boldsymbol{\gamma}}$, computes a valuation v^i such that $\mathrm{diff}(v^i, v^{i+1}) \leq \alpha$ within a number $i \in O(\boldsymbol{\gamma}^2)$ of iterations. Thus the limit function Lim can be defined as for probabilistic graphs, and guarantees that Algorithm 1 returns $\mathsf{supinf}\,\mathsf{Max}(p)$ (resp. $\mathsf{supinf}\,\mathsf{Min}(p)$).

Optimality. The maximizing and minimizing problems on probabilistic games lie in NP $\cap$ coNP. This was shown for boolean reachability objectives in [11], and the argument can be generalized to quantitative objectives. No polynomial-time algorithms are known for solving the maximizing and minimizing problems on probabilistic games, even in the special case of boolean reachability objectives. In particular, the linear-programming approach for probabilistic graphs does not generalize to probabilistic games.

3.7 Concurrent games

Concurrent strategies. Concurrent strategies are probabilistic. However, in concurrent games, the players choose distributions of moves, not of successor states. Let $G = (S, A_1, A_2, (E, \delta), D)$ be a concurrent game. For $k \in \{1, 2\}$, a *concurrent player-k strategy* is a function $\sigma : S^+ \to \mathrm{Dist}(A_k)$. We write Σ and Π for the sets of concurrent player-1 and player-2 strategies, respectively. Given two concurrent strategies $\sigma \in \Sigma$ and $\pi \in \Pi$, and a state $s \in S$, we write $\mathrm{Pr}_s^{\sigma,\pi}$ for the probability measure on the set Ω of paths which is defined as follows: for all $w \in S^*$ and all $t, t' \in S$, (1) $\mathrm{Pr}_s^{\sigma,\pi}(t \cdot S^\omega)$ is 1 if $t = s$, and it is 0 otherwise; and (2) if $\mathrm{Pr}_s^{\sigma,\pi}(w \cdot t \cdot S^\omega) = x$, then $\mathrm{Pr}_s^{\sigma,\pi}(w \cdot t \cdot t' \cdot S^\omega) = x \cdot \sum_{a_1 \in A_1} \sum_{a_2 \in A_2} \sigma(w \cdot t)(a_1) \cdot \pi(w \cdot t)(a_2) \cdot \delta(t, a_1, a_2)(t')$. This definition of $\mathrm{Pr}_s^{\sigma,\pi}$ for the basic open sets of paths suffices to obtain probabilities for all Borel sets.

Concurrent game values of objectives. On a concurrent game G, an objective W defines the valuation $\mathsf{supinf}\,W\colon S \to D$ given by

$$\mathsf{supinf}\,W(s) \;=\; \sup_{\sigma \in \Sigma} \inf_{\pi \in \Pi} \mathrm{E}_s^{\sigma,\pi}[W].$$

We refer to $\mathsf{supinf}\,W$ as the *concurrent game valuation* of the objective W. The maximizing and minimizing problems on concurrent games ask, given a concurrent game G and a reward function p, to compute the concurrent game valuations $\mathsf{supinf\,Max}(p)$ and $\mathsf{supinf\,Min}(p)$.

Concurrent game predecessor operator. The *concurrent game predecessor operator* $\mathsf{supinfPre}\colon V \to V$ is the function on valuations defined by

$$\begin{aligned}&\mathsf{supinfPre}(v)(s) \;=\\ &\sup_{\tau_1 \in \mathrm{Dist}(A_1)} \inf_{\tau_2 \in \mathrm{Dist}(A_2)} \sum_{s' \in S} \sum_{a_1 \in A_1} \sum_{a_2 \in A_2} v(s') \cdot \tau_1(s)(a_1) \cdot \tau_2(s)(a_2) \cdot \delta(s, a_1, a_2)(s');\end{aligned}$$

that is, the value of $\mathsf{supinfPre}(v)$ at a state s is the maximal value of v which player 1 can ensure at a successor of v, against all probabilistic choices of moves for player 2. In other words, the predecessor operator $\mathsf{supinfPre}$ solves a matrix game whose payoff function is specified by the valuation v. The function $\mathsf{supinfPre}$ is monotone, continuous and local.

Concurrent game valuations as improvement fixpoints. The concurrent game valuations of maximizing and minimizing objectives can be expressed as improvement fixpoints, but these fixpoints are not known to be finitely computable. Consider a reward function p, and the corresponding objectives $\mathsf{Max}(p)$ and $\mathsf{Min}(p)$. We redefine the two improvement functions:

$$\begin{aligned}\mathsf{maxImp}(v)(s) &= \max\{v(s), \mathsf{supinfPre}(v)(s)\};\\ \mathsf{minImp}(v)(s) &= \min\{v(s), \mathsf{supinfPre}(v)(s)\}.\end{aligned}$$

Note that maxImp is still extensive, and minImp reductive. The improvement fixpoint $\lim C(p, \mathsf{maxImp})$ for the initial valuation $v^0 = p$ and the improvement function maxImp is the concurrent game valuation $\mathsf{supinf\,Max}(p)$ of the maximizing objective, and the improvement fixpoint $\lim C(p, \mathsf{minImp})$ is the concurrent game valuation $\mathsf{supinf\,Min}(p)$ of the minimizing objective. This was shown for boolean objectives in [16], and the argument can be generalized to quantitative objectives. It is an open problem if the two improvement fixpoints are finitely computable. Indeed, even if all transition probabilities are rational, in general the values of boolean reachability objectives are irrational (but algebraic) [16]. This is in stark contrast to turn-based probabilistic games. For concurrent games, it is not even known if the improvement fixpoints are finitely approximable. More precisely, we do not know a time-bounded complexity class $\mathbb{C}$ such that for every real $\varepsilon > 0$ and every initial valuation v^0, there exists a function $f \in \mathbb{C}$ such that $\mathrm{diff}(v^i, v^\infty) \leq \varepsilon$ for $v^\infty = \lim C(p, \mathsf{maxImp})$ and $i = f(G, v^0, \varepsilon)$.

Optimality. The best known complexity bound for the maximizing and minimizing problems on concurrent games is EXPTIME. Specifically, since all improvement functions we consider can be defined in the theory of the reals with addition and multiplication, we can also express improvement fixpoints in this theory, which can be decided in EXPTIME. For boolean objectives a better bound is known: given a rational r in binary and an integer $d > 0$ in unary, there exists an NP Turing machine that is guaranteed to answer Yes if the concurrent game value of a boolean maximizing (or minimizing) objective at a state

$\boldsymbol{n}$ states	Objective $\mathsf{Max}(p)$	Objective $\mathsf{Min}(p)$
Valued graphs	$\mathsf{Imp}(v) = \max\{v, \mathsf{maxPre}(v)\}$ iteration converges in $\boldsymbol{n}$ steps	$\mathsf{Imp}(v) = \min\{v, \mathsf{maxPre}(v)\}$ iteration converges in $\boldsymbol{n}$ steps
Deterministic games	$\mathsf{Imp}(v) = \max\{v, \mathsf{maxminPre}(v)\}$ iteration converges in $\boldsymbol{n}$ steps	$\mathsf{Imp}(v) = \min\{v, \mathsf{maxminPre}(v)\}$ iteration converges in $\boldsymbol{n}$ steps
Probabilistic graphs	$\mathsf{Imp}(v) = \max\{v, \mathsf{maxPre}^P(v)\}$ iteration converges in $O(\boldsymbol{\gamma}^2)$ steps	$\mathsf{Imp} = \min\{v, \mathsf{maxPre}^P(v)\}$ iteration converges in $O(\boldsymbol{\gamma}^2)$ steps
Probabilistic games	$\mathsf{Imp}(v) = \max\{v, \mathsf{maxminPre}^P(v)\}$ iteration converges in $O(\boldsymbol{\gamma}^2)$ steps	$\mathsf{Imp}(v) = \min\{v, \mathsf{maxminPre}^P(v)\}$ iteration converges in $O(\boldsymbol{\gamma}^2)$ steps
Concurrent games	$\mathsf{Imp}(v) = \max\{v, \mathsf{supinfPre}(v)\}$ iteration converges in the limit (no known bound)	$\mathsf{Imp}(v) = \min\{v, \mathsf{supinfPre}(v)\}$ iteration converges in the limit (no known bound)

Table 1. Value improvement for maximizing and minimizing objectives. Recall that $\boldsymbol{\gamma}$ is such that $16^{\boldsymbol{n}} \in O(\boldsymbol{\gamma})$.

is greater than $r + \frac{1}{d}$, and No if it is less than $r - \frac{1}{d}$. This was shown (but misstated) in [4]. However, the argument does not generalize to quantitative objectives.

Summary. We summarize the situation for maximizing and minimizing objectives in Table 1.

4 Level-2 Objectives and Depth-1 Value Iteration

Maximizing and minimizing objectives are obtained within a finite number of transitions. The simplest kind of infinite objectives are boolean Büchi and coBüchi objectives. In the quantitative case, these Borel level-2 objectives generalize to limsup and liminf objectives.

4.1 Limsup and liminf objectives

Consider a valued graph (S, E, D) and a reward function $p: S \to D$. The *limsup objective* $\mathsf{LimSup}(p): \Omega \to D$ is the function that maps every path to the maximal reward appearing infinitely often along the path. Formally, for all paths $\omega = \langle s_0, s_1, s_2, \ldots \rangle$,

$$\mathsf{LimSup}(p)(\omega) \;=\; \lim_{n\to\infty} \max\{p(s_i) \mid i \geq n\}.$$

Observe that $C = \langle p_0, p_1, p_2, \ldots \rangle$, where $p_n = \max\{p(s_i) \mid i \geq n\}$ for all $n \geq 0$, is a decreasing sequence (chain) of values, and its limit is the greatest lower bound of C. In the boolean case, limsup objectives are Büchi objectives; they require a path to visit a *recurrent* set p infinitely often: we have $\omega \in \mathsf{LimSup}(p)$ iff $s_i \in p$ for infinitely many $i \geq 0$. The *liminf objective* $\mathsf{LimInf}(p): \Omega \to D$ is defined dually, by

$$\mathsf{LimInf}(p)(\omega) \;=\; \lim_{n\to\infty} \min\{p(s_i) \mid i \geq n\};$$

Algorithm 2 NestedValueImprovement

Input: valued graph G, binary improvement function $\mathsf{Imp2}$, limit function Lim, precision $\alpha \in \mathbb{R}_{\geq 0}$, and two initial valuations $v^0, u^0 \in V$.
Output: valuation $v^* \in V$.

$i := 0$;
do {
  $v^{i+1} := \text{innerLoop}(v^i, u^0)$;
  $i := i + 1$;
  } until $\text{diff}(v^{i-1}, v^i) \leq \alpha$;
return $v^* := \mathsf{Lim}(v^i, \alpha)$.

procedure innerLoop(v^i, u_i^0):
  $j := 0$;
  do {
    $u_i^{j+1} := \mathsf{Imp2}(v^i, u_i^j)$;
    $j := j + 1$;
    } until $\text{diff}(u_i^{j-1}, u_i^j) \leq \alpha$;
  return $u_i^* := \mathsf{Lim}(u_i^j, \alpha)$.

---

that is, $\mathsf{LimInf}(p)$ is a least upper bound. Boolean LimInf objectives are coBüchi objectives; they require a path to eventually stay in a *persistent* set p: we have $\omega \in \mathsf{LimInf}(p)$ iff $s_i \in p$ for all but finitely many $i \geq 0$. While the boolean $\mathsf{LimSup}(p)$ objective corresponds to the formula $\Box\Diamond p$ of linear temporal logic, the boolean $\mathsf{LimInf}(p)$ objective corresponds to the formula $\Diamond\Box p$. Both limsup and liminf objectives lie on level 2 of the Borel hierarchy.

Limsup and liminf problems. Given a valued graph G (resp. a deterministic game; a probabilistic graph; a probabilistic game; or a concurrent game), and a reward function p, we wish to compute the valuations $\mathsf{sup}\,\mathsf{LimSup}(p)$ and $\mathsf{sup}\,\mathsf{LimInf}(p)$ over G. Note that the graph valuation of the boolean $\mathsf{LimSup}(p)$ objective corresponds to the formula $\exists\Box\Diamond p$ of branching temporal logic; and the graph valuation of the boolean $\mathsf{LimInf}(p)$ objective corresponds to the formula $\exists\Diamond\Box p$. Hence in the boolean case, the limsup and liminf problems on graphs (resp. games) arise in model checking CTL (resp. ATL) over structures with weak-fairness (Büchi) constraints [10]. Also the model-checking problem for the linear temporal logic LTL can be reduced to the boolean limsup problem on graphs (or probabilistic graphs), by converting the negation of a given LTL formula into a nondeterministic Büchi automaton of exponential size [13, 10].

4.2 Nested value improvement

We refer to value iteration schemes of alternation depth-1 as *nested value improvement*. The nested value improvement algorithm operates on a valued graph

$G = (S, E, D)$ using a binary improvement function $\mathsf{Imp2}$ and a limit function Lim. A binary improvement function maps a pair of valuations to a new valuation.

Binary improvement functions. The valuation pairs $V \times V$ form a complete lattice —the product lattice— with the following ordering: for two valuation pairs $(v_1, u_1), (v_2, u_2) \in V \times V$, let $(v_1, u_1) \leq (v_2, u_2)$ iff both $v_1 \leq v_2$ and $u_1 \leq u_2$. Thus all infinite increasing and decreasing sequences (chains) of valuation pairs have limits. Every chain $C = \langle (v_0, u_0), (v_1, u_1), (v_2, u_2), \ldots \rangle$ of valuation pairs consists of two chains $C_1 = \langle v_0, v_1, v_2, \ldots \rangle$ and $C_2 = \langle u_0, u_1, u_2, \ldots \rangle$ of valuations; note that $\lim C = (\lim C_1, \lim C_2)$. A *binary improvement function* $\mathsf{Imp2}$: $V \times V \to V$ is a function on valuation pairs which satisfies the following requirements.

Monotone For all valuation pairs $(v_1, u_1), (v_2, u_2) \in V \times V$, if $(v_1, u_1) \leq (v_2, u_2)$, then $\mathsf{Imp2}(v_1, u_1) \leq \mathsf{Imp2}(v_2, u_2)$.

Continuous For every chain $C = \langle (v_0, u_0), (v_1, u_1), (v_2, u_2), \ldots \rangle$ of valuation pairs, the sequence $\mathsf{Imp2}(C) = \langle \mathsf{Imp2}(v_0, u_0), \mathsf{Imp2}(v_1, u_1), \mathsf{Imp2}(v_2, u_2), \ldots \rangle$ is a chain of valuations because of the monotonicity of $\mathsf{Imp2}$. We require that $\mathsf{Imp2}(\lim C) = \lim \mathsf{Imp2}(C)$.

Directed Either $v \leq \mathsf{Imp2}(v, u) \leq u$ for all valuations $v, u \in V$ with $v \leq u$; or $v \geq \mathsf{Imp2}(v, u) \geq u$ for all valuations $v, u \in V$ with $v \geq u$.

The binary improvement functions we consider also satisfy the following *locality* property: for all states $s \in S$ and all valuation pairs $(v_1, u_1), (v_2, u_2) \in V \times V$, if $v_1(s') = v_2(s')$ and $u_1(s') = u_2(s')$ for all successors $s' \in E(s)$, then $\mathsf{Imp2}(v_1, u_1)(s) = \mathsf{Imp2}(v_2, u_2)(s)$.

The nested value improvement algorithm. The nested value improvement algorithm (Algorithm 2) takes as input a valued graph, a binary improvement function $\mathsf{Imp2}$, a limit function Lim, a precision $\alpha \in \mathbb{R}_{\geq 0}$, an *outer* initial valuation $v^0 \in V$, and an *inner* initial valuation $u^0 \in V$. The algorithm returns a valuation $v^* \in V$ that lies between v^0 and u^0 with respect to the lattice ordering of valuations. It suffices to choose the initial valuations v^0 and u^0 such that the desired result v^* lies between v^0 and u^0. Thus, it is possible to choose either v^0 to be the bottom element of the lattice, and u^0 the top, or vice versa. Alternatively, in our uses of the algorithm, we can always choose one of them to be $\min p$ and the other one $\max p$, where $(\min p)(s) = \min\{p(t) \mid t \in S\}$ and $(\max p)(s) = \max\{p(t) \mid t \in S\}$ for all states $s \in S$. This is because the values of the rewards that appear infinitely often along a path, or that appear all but finitely often along a path, lie between $\min p$ and $\max p$. However, the result v^* of the algorithm depends on the ordering of the two initial valuations; that is, it is important whether the outer initial valuation is less than the inner initial valuation (case $v^0 \leq u^0$), or greater (case $v^0 \geq u^0$).

Starting from the outer initial valuation v^0, the algorithm iteratively improves the valuation in order to compute the limit v^∞ of an *outer improvement chain* $C_1(v^0, u^0, \mathsf{Imp2}) = \langle v^0, v^1, v^2, \ldots \rangle$ of valuations. Each outer improvement step is itself the result of computing the limit of an inner improvement chain:

we have $v^{i+1} = u_i^\infty = \lim C_2^i(v^i, u^0, \mathsf{Imp2})$ for all $i \geq 0$. For each $i \geq 0$, the i-th *inner improvement chain* $C_2^i(v^i, u^0, \mathsf{Imp2}) = \langle u_i^0, u_i^1, u_i^2, \ldots \rangle$ of valuations results from iteratively applying the directed improvement function $\mathsf{Imp2}$ to the inner initial valuation u^0: we have $u_i^0 = u^0$, and $u_i^{j+1} = \mathsf{Imp2}(v^i, u_i^j)$ for all $j \geq 0$. In other words,

$$v^\infty \;=\; \lim_{i\to\infty} u_i^\infty \;=\; \lim_{i\to\infty} \lim_{j\to\infty} u_i^j.$$

In case $v^0 \leq u^0$, since $\mathsf{Imp2}$ is directed, for all $i \geq 0$, the inner improvement chain $C_2^i(v^i, u^0, \mathsf{Imp2})$ is decreasing. Since $\mathsf{Imp2}$ is directed, we also have $v^i \leq u_i^j$ for all $i, j \geq 0$. Hence $v^i \leq u_i^\infty = v^{i+1}$, and thus the outer improvement chain $C_1(v^0, u^0, \mathsf{Imp2})$ is increasing. On the other hand, in case $v^0 \geq u^0$, all inner improvement chains $C_2^i(v^i, u^0, \mathsf{Imp2})$ are increasing, and the outer improvement chain $C_1(v^0, u^0, \mathsf{Imp2})$ is decreasing. Observe that, as $\mathsf{Imp2}$ is monotone, also $u_i^j \leq u_{i+1}^j$ for all $i, j \geq 0$ if $v^0 \leq u^0$, and $u_i^j \geq u_{i+1}^j$ for all $i, j \geq 0$ if $v^0 \geq u^0$.

If successful, the algorithm returns the limit v^∞ of the outer improvement chain. As in the alternation-free case, success means either that all limits u_i^∞, for $i \geq 0$, and v^∞ are finitely reachable, or that they are finitely computable using the precision α and the limit function Lim for acceleration. If finite computability fails, then we ask the question of finite approximability of the outer limit v^∞.

Fixpoint characterization. We consider first the case $v^0 \leq u^0$. For all $i \geq 0$, the limit u_i^∞ of the i-th inner improvement chain $C_2^i(v^i, u^0, \mathsf{Imp2})$ is the greatest fixpoint below u^0 of the monotone and continuous function $\mathsf{Imp2}(v^i, \cdot)\colon V \to V$. The limit v^∞ of the outer improvement chain $C_1(v^0, u^0, \mathsf{Imp2})$ is the least fixpoint above v^0 of the function $f\colon V \to V$, which is defined by $f(v) = \mathrm{lub}\{u \in V \mid u \leq u^0 \text{ and } \mathsf{Imp2}(v, u) = u\}$ for all valuations $v \in V$. Note that f is again monotone and continuous. Thus, in this case, the outer limit v^∞ is the least fixpoint of a function whose values are greatest fixpoints. In μ-calculus notation,

$$v^\infty \;=\; (\mu X \geq v^0)(\nu Y \leq u^0)\ \mathsf{Imp2}(X, Y).$$

The case $v^0 \geq u^0$ is symmetric. In this case, the outer limit v^∞ is the greatest fixpoint of a function whose values are least fixpoints, namely,

$$v^\infty \;=\; (\nu X \leq v^0)(\mu Y \geq u^0)\ \mathsf{Imp2}(X, Y).$$

We henceforth refer to v^∞ as *outer improvement fixpoint*, and to u_i^∞ as i-th *inner improvement fixpoint*, for $i \geq 0$.

Parametric improvement functions. We define binary improvement functions with a parameter Pre that, for different classes of graph models, will be instantiated by different predecessor operators. Consider a reward function p, and the corresponding objectives $\mathsf{LimSup}(p)$ and $\mathsf{LimInf}(p)$. Given a function $\mathsf{Pre}\colon V \to V$, we define the two parametric functions $\mathsf{limsupImp}[\mathsf{Pre}]\colon V \times V \to V$ and $\mathsf{liminfImp}[\mathsf{Pre}]\colon V \times V \to V$ by

$$\begin{aligned}
\mathsf{limsupImp}[\mathsf{Pre}](v, u) &= \min\{\max\{p, u, \mathsf{Pre}(u)\}, v, \max\{u, \mathsf{Pre}(v)\}\};\\
\mathsf{liminfImp}[\mathsf{Pre}](v, u) &= \max\{\min\{p, u, \mathsf{Pre}(u)\}, v, \min\{u, \mathsf{Pre}(v)\}\};
\end{aligned}$$

for all valuations $v, u \in V$ (the functions max and min are lifted from values to valuations in a pointwise fashion). Observe that if $v \geq u$, then $v \geq \mathsf{limsupImp}[\mathsf{Pre}](v, u) \geq u$; and if $v \leq u$, then $v \leq \mathsf{liminfImp}[\mathsf{Pre}](v, u) \leq u$. Thus both $\mathsf{limsupImp}[\mathsf{Pre}]$ and $\mathsf{liminfImp}[\mathsf{Pre}]$ are directed. For different graph models, we will instantiate the parameter Pre by one of the predecessor operators maxPre, $\mathsf{maxminPre}$, maxPre^P, $\mathsf{maxminPre}^P$, or $\mathsf{supinfPre}$ from Section 3. It should be remarked that in the cases we consider, we can simplify the definitions of the binary improvement functions as follows:

$$\begin{aligned}\mathsf{limsupImp}[\mathsf{Pre}](v, u) &= \min\{\max\{p, u, \mathsf{Pre}(u)\}, v, \mathsf{Pre}(v)\};\\ \mathsf{liminfImp}[\mathsf{Pre}](v, u) &= \max\{\min\{p, u, \mathsf{Pre}(u)\}, v, \mathsf{Pre}(v)\};\end{aligned}$$

for all valuations $v, u \in V$. To see why the simplification works, let $u^{j+1} = \mathsf{limsupImp}[\mathsf{Pre}](v, u^j)$ (according to the original, unsimplified definition) for all $j \geq 0$. For all valuations $v \geq u^0$, if $\mathsf{Pre}(v) \geq u^0$, then for all $j \geq 0$, both $v \geq u^j$ and $\mathsf{Pre}(v) \geq u^j$, and therefore $u^{j+1} = \min\{\max\{p, u^j, \mathsf{Pre}(u^j)\}, v, \mathsf{Pre}(v)\}$. If $u^0 = \min p$, and Pre is one of maxPre, $\mathsf{maxminPre}$, maxPre^P, $\mathsf{maxminPre}^P$, or $\mathsf{supinfPre}$, then for all valuations $v \geq u^0$, we have $\mathsf{Pre}(v) \geq u^0$, and thus the above simplification works. The case $u^0 = \max p$ and $\mathsf{liminfImp}[\mathsf{Pre}]$ is symmetric.

4.3 Graphs

Finitely reachable nested fixpoints. The graph valuations of limsup and liminf objectives can be computed by nested value improvement. On a valued graph $G = (S, E, D)$, the outer improvement fixpoint $\lim C_1(v^0, u^0, \mathsf{Imp2})$, for the outer initial valuation $v^0 = \max p$, the inner initial valuation $u^0 = \min p$, and the improvement function $\mathsf{Imp2} = \mathsf{limsupImp}[\mathsf{maxPre}]$, is the graph valuation $\sup \mathsf{LimSup}(p)$ of the limsup objective. Similarly, the outer improvement fixpoint $\lim C_1(\min p, \max p, \mathsf{liminfImp}[\mathsf{maxPre}])$ is the graph valuation $\sup \mathsf{LimInf}(p)$ of the liminf objective. Each inner improvement fixpoint is finitely reachable within at most $\boldsymbol{n}$ steps, and the outer improvement fixpoints are finitely reachable within at most $\boldsymbol{n}$ computations of inner improvement fixpoints. Hence the total number of applications of the improvement function $\mathsf{Imp2}$ in Algorithm 2 is bounded by $\boldsymbol{n}^2$ when computing $\sup \mathsf{LimSup}(p)$ or $\sup \mathsf{LimInf}(p)$ on graphs.

Optimality. Every improvement step (i.e., each application of the function $\mathsf{limsupImp}$ or $\mathsf{liminfImp}$) can be computed in $O(\boldsymbol{m})$ time. Hence a direct implementation of Algorithm 2 has the time complexity $O(\boldsymbol{m}\boldsymbol{n}^2)$. In the boolean case, the nested value improvement scheme can be sped up to run in $O(\boldsymbol{m}\boldsymbol{n})$ time by computing inner improvement fixpoints using techniques similar to the $O(\boldsymbol{m})$ implementations of Max and Min objectives. Yet, unlike in the case of maximizing and minimizing objectives, the nested value improvement scheme is not known to have an optimal implementation even in the boolean case. This is because the graph valuations of Büchi and coBüchi objectives (i.e., boolean limsup and liminf objectives) can be computed in $O(\boldsymbol{m})$ time by finding the maximal strongly connected components of a graph. In the quantitative case, the limsup and liminf problems on graphs can be solved in $O(\boldsymbol{m} + \boldsymbol{n} \log \boldsymbol{n})$

time by sorting the rewards of all states, computing maximal strongly connected components, and applying the algorithms for Max (resp. Min) objectives in descending (resp. ascending) order of rewards. We know of no implementation of the nested value improvement scheme which matches this complexity.

4.4 Deterministic games

Finitely reachable nested fixpoints. The deterministic game valuations of limsup and liminf objectives can again be computed by nested value improvement, using a different predecessor operator. On a deterministic game $G = ((S, S_1, S_2), E, D)$, the outer improvement fixpoint $\lim C_1(\max p, \min p, \mathsf{limsupImp}[\mathsf{maxminPre}])$ is the deterministic game valuation $\mathsf{supinf}\,\mathsf{LimSup}(p)$ of the limsup objective, and $\lim C_1(\min p, \max p, \mathsf{liminfImp}[\mathsf{maxminPre}])$ is the deterministic game valuation $\mathsf{supinf}\,\mathsf{LimInf}(p)$ of the liminf objective. Each inner improvement fixpoint is finitely reachable within at most $\boldsymbol{n}$ steps, and the outer improvement fixpoints are finitely reachable within at most $\boldsymbol{n}$ computations of inner improvement fixpoints. Hence, as in the case of graphs, the total number of applications of the improvement function $\mathsf{Imp2}$ in Algorithm 2 is bounded by $\boldsymbol{n}^2$ when computing $\mathsf{supinf}\,\mathsf{LimSup}(p)$ or $\mathsf{supinf}\,\mathsf{LimInf}(p)$ on deterministic games.

Example 5 (Deterministic game with limsup objective). Consider the deterministic game shown in Figure 4, where the reward function p is indicated by state labels. We consider the objective $\mathsf{LimSup}(p)$ for player 1 (the $\Box$ player). We specify valuations as value vectors as before; the outer initial valuation is $v^0 = \langle 15, 15, 15, 15, 15\rangle$, and the inner initial valuation is $u^0 = \langle 5, 5, 5, 5, 5\rangle$. We compute the first inner improvement fixpoint: $u_0^0 = \langle 5, 5, 5, 5, 5\rangle$, and since

$$u_0^{j+1} \;=\; \min\{\max\{p, u_0^j, \mathsf{maxminPre}(u_0^j)\}, v^0, \mathsf{maxminPre}(v^0)\}$$

for all $j \geq 0$, where $v^0 = \mathsf{maxminPre}(v^0) = \langle 15, 15, 15, 15, 15\rangle$, we obtain $u_0^1 = \langle 5, 5, 15, 10, 5\rangle$. Note that u_0^1 coincides with the reward function p. Next we obtain $u_0^2 = \langle 10, 5, 15, 10, 10\rangle$, because $\max\{p, u_0^1, \mathsf{maxminPre}(u_0^1)\} = \langle 10, 5, 15, 10, 10\rangle$. Finally $u_0^3 = u_0^4 = \langle 10, 10, 15, 10, 10\rangle$, which is the first inner improvement fixpoint v^1. The second inner improvement chain starts with $u_1^0 = \langle 5, 5, 5, 5, 5\rangle$ using

$$u_1^{j+1} \;=\; \min\{\max\{p, u_1^j, \mathsf{maxminPre}(u_1^j)\}, v^1, \mathsf{maxminPre}(v^1)\},$$

where $v^1 = \langle 10, 10, 15, 10, 10\rangle$ and $\mathsf{maxminPre}(v^1) = \langle 10, 10, 10, 10, 10\rangle$. Since $\max\{p, u_1^0, \mathsf{maxminPre}(u_1^0)\} = \langle 5, 5, 15, 10, 5\rangle$, we obtain $u_1^2 = \langle 10, 5, 10, 10, 10\rangle$. Then $u_1^3 = u_1^4 = \langle 10, 10, 10, 10, 10\rangle$, which is the second inner improvement fixpoint v^2. This is also the desired outer improvement fixpoint; that is, $v^\infty = v^2 = v^3 = \langle 10, 10, 10, 10, 10\rangle$. The player-1 strategy that chooses at state s_0 the successor s_3 ensures that against all strategies of player 2, the reward 10 will be visited infinitely often. Dually, the player-2 strategy that chooses at s_1 the successor s_0 ensures that against all strategies of player 1, the reward 15

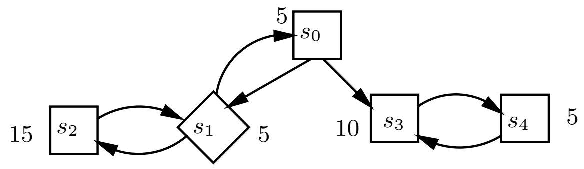

Fig. 4. Deterministic game with limsup objective.

will be visited at most once. Hence $\langle 10, 10, 10, 10, 10\rangle$ is the deterministic game valuation of the player-1 objective $\mathsf{LimSup}(p)$: from any start state, player 1 can ensure that reward 10 will be visited infinitely often, but she cannot do so for reward 15. □

Optimality. The situation is similar to the case of graphs. The direct implementation of nested value improvement yields a $O(\boldsymbol{mn}^2)$ time complexity. In the boolean case, the scheme can be sped up to run in $O(\boldsymbol{mn})$ time by using $O(\boldsymbol{m})$ implementations for Max and Min objectives to compute inner improvement fixpoints. However, unlike in the case of graphs, $O(\boldsymbol{mn})$ is the best known time bound for solving deterministic games with Büchi or coBüchi objectives; that is, nested value improvement for deterministic games with boolean objectives is optimal. In the quantitative case, the limsup and liminf problems on deterministic games can be solved in $O((\boldsymbol{m} + \log \boldsymbol{n}) \cdot \boldsymbol{n})$ time [2], but we know of no implementation of nested value improvement which matches this complexity.

4.5 Probabilistic graphs

Finitely computable nested fixpoints. The probabilistic graph valuations of limsup and liminf objectives can be expressed as nested improvement fixpoints; while these fixpoints are not finitely reachable, they are finitely computable. On a probabilistic graph $G = ((S, S_1, S_*), (E, \delta), D)$, the outer improvement fixpoint $\lim C_1(\max p, \min p, \mathsf{limsupImp}[\mathsf{maxPre}^P])$ is the probabilistic graph valuation $\mathsf{sup}\,\mathsf{LimSup}(p)$ of the limsup objective, and $\lim C_1(\min p, \max p, \mathsf{liminfImp}[\mathsf{maxPre}^P])$ is the probabilistic graph valuation $\mathsf{sup}\,\mathsf{LimInf}(p)$ of the liminf objective. Neither the inner nor the outer improvement fixpoints are finitely reachable. However, the boundedness property of probabilistic graph values for maximizing and minimizing objectives carries over to limsup and liminf objectives. This is the key for proving the finite computability: the nested value improvement algorithm (Algorithm 2) with precision $\alpha = \frac{1}{2\boldsymbol{\gamma}}$, where $\boldsymbol{\gamma}$ is defined as in Section 3, achieves the following: (1) in the outer loop, Algorithm 2 computes a valuation v^i such that $\mathrm{diff}(v^i, v^{i+1}) \leq \alpha$ within a number $i \in O(\boldsymbol{\gamma}^2)$ of iterations; and (2) in the inner loop, for each $i \geq 0$, Algorithm 2 computes a valuation u_i^j such that $\mathrm{diff}(u_i^j, u_i^{j+1}) \leq \alpha$ within a number $j \in O(\boldsymbol{\gamma}^2)$ of iterations. Thus the limit function Lim to obtain v^* from v^i, and each u_i^* from u_i^j, can be defined as follows: for every state $s \in S$, round the value $v^i(s)$

(resp. $u_i^j(s)$) to the nearest multiple of $\frac{1}{\gamma}$. Then for each $i \geq 0$, the accelerated valuation u_i^* is the i-th inner improvement fixpoint u_i^∞, and v^* is the outer improvement fixpoint v^∞.

Optimality. While no better bound than an exponential bound in the number of states is known for the nested value improvement scheme, the probabilistic graph valuations of limsup and liminf objectives can be computed in polynomial time. In the boolean case, for Büchi and coBüchi objectives, the polynomial-time computation proceeds as follows: first compute the *value-1* set $T \subseteq S$ of the given Büchi or coBüchi objective (i.e., the set of states s such that the probabilistic graph valuation at s is 1), and then compute the probabilistic graph valuation of the reachability objective with target set T; the latter values can be computed by linear programming (see Section 3 on maximizing and minimizing objectives). A polynomial-time algorithm for computing the value-1 set T for Büchi and coBüchi objectives is presented in [14, 8]. In the quantitative case, the probabilistic graph valuations of limsup and liminf objectives can be obtained by first computing the value-1 sets for Büchi (resp. coBüchi) objectives in descending (resp. ascending) order of the rewards, and then using the linear-programming approach to compute the probabilistic graph valuations for maximizing (resp. minimizing) objectives. As pointed out in Section 3, these techniques based on linear programming do not generalize to probabilistic games, even in the special case of boolean objectives.

4.6 Probabilistic games

Finitely computable nested fixpoints. The probabilistic game valuations of limsup and liminf objectives can again be expressed as nested improvement fixpoints: on a probabilistic game $G = ((S, S_1, S_2, S_*), (E, \delta), D)$, the outer improvement fixpoint $\lim C_1(\max p, \min p, \mathsf{limsupImp}[\mathsf{maxminPre}^P])$ is the probabilistic game valuation $\mathsf{supinf}\,\mathsf{LimSup}(p)$ of the limsup objective, and $\lim C_1(\min p, \max p, \mathsf{liminfImp}[\mathsf{maxminPre}^P])$ is the probabilistic game valuation $\mathsf{supinf}\,\mathsf{LimInf}(p)$ of the liminf objective. The boundedness property and finite-computability results for limsup and liminf objectives on probabilistic graphs carry over to probabilistic games; see Table 2.

Optimality. Unlike in the case of probabilistic graphs, no polynomial-time algorithms are known for solving the limsup and liminf problems on probabilistic games, not even in the special case of boolean (Büchi and coBüchi) objectives. The problems of computing probabilistic game valuations for general (quantitative) limsup and liminf objectives can be shown to lie in NP $\cap$ coNP.

4.7 Concurrent games

Nested improvement fixpoints. The concurrent game valuations of Büchi and coBüchi objectives can be expressed as nested improvement fixpoints, but these fixpoints are not known to be finitely computable.

$\boldsymbol{n}$ states	Objective $\mathsf{LimSup}(p)$	Objective $\mathsf{LimInf}(p)$
Valued graphs	$\mathsf{Imp2}(v,u) = \mathsf{limsupImp}[\mathsf{maxPre}](v,u)$ iteration converges in $\boldsymbol{n}^2$ steps	$\mathsf{Imp2}(v,u) = \mathsf{liminfImp}[\mathsf{maxPre}](v,u)$ iteration converges in $\boldsymbol{n}^2$ steps
Deterministic games	$\mathsf{Imp2}(v,u) = \mathsf{limsupImp}[\mathsf{maxminPre}](v,u)$ iteration converges in $\boldsymbol{n}^2$ steps	$\mathsf{Imp2}(v,u) = \mathsf{liminfImp}[\mathsf{maxminPre}](v,u)$ iteration converges in $\boldsymbol{n}^2$ steps
Probabilistic graphs	$\mathsf{Imp2}(v,u) = \mathsf{limsupImp}[\mathsf{maxPre}^P](v,u)$ iteration converges in $O(\boldsymbol{\gamma}^4)$ steps	$\mathsf{Imp2}(v,u) = \mathsf{liminfImp}[\mathsf{maxPre}^P](v,u)$ iteration converges in $O(\boldsymbol{\gamma}^4)$ steps
Probabilistic games	$\mathsf{Imp2}(v,u) = \mathsf{limsupImp}[\mathsf{maxminPre}^P](v,u)$ iteration converges in $O(\boldsymbol{\gamma}^4)$ steps	$\mathsf{Imp2}(v,u) = \mathsf{liminfImp}[\mathsf{maxminPre}^P](v,u)$ iteration converges in $O(\boldsymbol{\gamma}^4)$ steps
Concurrent games (boolean p)	$\mathsf{Imp2}(v,u) = \mathsf{limsupImp}[\mathsf{supinfPre}](v,u)$ iteration converges in the limit (no known bound)	$\mathsf{Imp2}(v,u) = \mathsf{liminfImp}[\mathsf{supinfPre}](v,u)$ iteration converges in the limit (no known bound)

Table 2. Nested value improvement for limsup and liminf objectives. Recall that $\boldsymbol{\gamma}$ is such that $16^{\boldsymbol{n}} \in O(\boldsymbol{\gamma})$.

On a concurrent game $G = (S, A_1, A_2, (E, \delta), D)$, the outer improvement fixpoint $\lim C_1(\max p, \min p, \mathsf{limsupImp}[\mathsf{supinfPre}])$ is the concurrent game valuation $\mathsf{supinf}\,\Box\Diamond p$ of the boolean limsup (Büchi) objective, and $\lim C_1(\min p, \max p, \mathsf{liminfImp}[\mathsf{supinfPre}])$ is the concurrent game valuation $\mathsf{supinf}\,\Diamond\Box p$ of the boolean liminf (coBüchi) objective [16]. Neither the inner nor outer improvement fixpoints are known to be finitely computable. As in the case of reachability and safety objectives, it is not even known if any improvement fixpoints are finitely approximable. The quantitative case has not been studied in the literature: we conjecture that the above characterizations of the concurrent game valuations for Büchi and coBüchi objectives generalize to quantitative limsup and liminf objectives.

Optimality. The complexity results for concurrent games with reachability and safety objectives (see Section 3) generalize to Büchi and coBüchi objectives.

Summary. We summarize the situation for limsup and liminf objectives in Table 2.

5 Level-3 Objectives

We briefly discuss the most important objectives above Borel level 2: parity objectives and limit-average objectives. The parity objectives are a canonical form to express all ω-regular objectives, and the limit-average (often called *mean-payoff*) objectives are studied widely in game theory. For parity objectives, the value lattice is a finite linear order (or alternatively, a finite product of two-valued boolean lattices); for limit-average objectives, the value lattice is quantitative.

Parity objectives Let $D = \{0, 1, \ldots, \boldsymbol{d}-1\}$ or $D = \{1, 2, \ldots, \boldsymbol{d}\}$; the $\boldsymbol{d}$ integers in D are called *priorities*. Consider a valued graph (S, E, D) and a reward function p: $S \to D$. The *parity objective* $\mathsf{Parity}(p)$: $\Omega \to \mathbb{B}$ is the function that maps a path to 1 if the maximal priority that appears along the path infinitely often is even. Formally,

$$\mathsf{Parity}(p) \;=\; \{\omega \in \Omega \mid \mathsf{LimSup}(p)(\omega) \text{ is even}\}.$$

Büchi and coBüchi objectives are special cases of parity objectives with two priorities: for Büchi objectives, let $D = \{1, 2\}$, and $p(s) = 2$ iff s is in the recurrent set; for coBüchi objectives, let $D = \{0, 1\}$, and $p(s) = 0$ iff s is in the persistent set.

Limit-average objectives Let $D = \mathbb{R}^{\infty}_{\geq 0}$. Consider a valued graph (S, E, D) and a reward function p: $S \to D$. The *limit-average objective* $\mathsf{LimAvg}(p)$: $\Omega \to D$ is the function that maps every path to the long-run average of the rewards that appear along the path. Formally, for all paths $\omega = \langle s_0, s_1, s_2, \ldots \rangle$,

$$\mathsf{LimAvg}(p)(\omega) \;=\; \lim_{n\to\infty} \inf\{\frac{1}{k} \cdot \sum_{i=0}^{k-1} p(s_i) \mid k \geq n\}.$$

Borel complexity. The parity objectives lie in the intersection of the third levels of the Borel hierarchy, in $\Sigma_3 \cap \Pi_3$; they are hard for the second levels, i.e., both Σ_2-hard and Π_2-hard [31]. The limit-average objectives are Π_3-complete: for every real $\beta \geq 0$, the set of paths $\omega \in \Omega$ with $\mathsf{LimAvg}(p)(\omega) \geq \beta$ is definable in Π_3 by

$$(\forall m \geq 1)(\exists n \geq 0)(\forall k \geq n)(\frac{1}{k} \cdot \sum_{i=0}^{k-1} p(s_i) \geq \beta - \frac{1}{m});$$

and the set of paths $\omega \in \Omega$ with $\mathsf{LimAvg}(p)(\omega) \leq \beta$ is definable in Π_2 by

$$(\forall n \geq 0)(\exists k \geq n)(\frac{1}{k} \cdot \sum_{i=0}^{k-1} p(s_i) \leq \beta).$$

The Π_3-hardness of limit-average objectives is proved in [3].

Parity and limit-average problems. Given a valued graph G (resp. a deterministic game; a probabilistic graph; a probabilistic game; or a concurrent game), and a reward function p, we wish to compute the graph valuations $\mathsf{sup}\,\mathsf{Parity}(p)$ and $\mathsf{sup}\,\mathsf{LimAvg}(p)$ (resp. the game valuations $\mathsf{supinf}\,\mathsf{Parity}(p)$ and $\mathsf{supinf}\,\mathsf{LimAvg}(p)$) over G. The parity problem on graphs (resp. games) arises in model checking CTL (resp. ATL) over structures with strong-fairness (Streett) constraints, which can be converted to parity constraints. Also the synthesis problem for LTL can be reduced to a parity problem on games, by converting a given LTL formula into a deterministic parity automaton of double-exponential size [29].

5.1 Parity objectives

Value iteration solutions for parity (and limit-average) objectives were proposed first for deterministic games; so we start with this case.

Deterministic games. The parity problem on deterministic games is equivalent to μ-calculus model checking [17]: given a value set with $\boldsymbol{d}$ priorities, the deterministic game valuation $\sup \mathsf{Parity}(p)$ can be computed by evaluating a μ-calculus expression over the deterministic game predecessor operator $\mathsf{maxminPre}$. The μ-calculus expression has alternation depth $\boldsymbol{d}-1$, that is, it contains $\boldsymbol{d}-1$ alternations of the least-fixpoint operator μ and the greatest-fixpoint operator ν (the μ-calculus formulas for Büchi and coBüchi objectives from Section 4, which have alternation depth 1, are obtained as special cases for $\boldsymbol{d}=2$). The μ-calculus expression defines a value iteration scheme that computes $\boldsymbol{d}$ nested fixpoints. All fixpoints are finitely reachable, and thus the value iteration algorithm runs in $O(\boldsymbol{m}\boldsymbol{n}^{\boldsymbol{d}-1})$ time.

For the value iteration scheme obtained from the μ-calculus expression of [17], the values are boolean. In [21], a different value lattice is considered, namely, tuples of nonnegative integers with lexicographic ordering. A valuation assigns to each state a so-called *rank* (or *small-progress measure*), which is a $\boldsymbol{d}$-tuple such that every even coordinate is 0, and every odd coordinate is an integer between 0 and $\boldsymbol{n}$. For such "rich" valuations, an alternation-free value iteration scheme (similar to Algorithm 1) can be used to solve parity games. The value improvement fixpoint is finitely reachable and can be computed in $O(\boldsymbol{m}\boldsymbol{n}^{\lfloor \boldsymbol{d}/2 \rfloor})$ time [21]. It is an interesting question when and how in general the complexity of a value iteration scheme can be traded off against the complexity of a modified value domain. If $\boldsymbol{d} \leq \sqrt{\boldsymbol{n}}$, then [21] is the best known algorithm. In [22], a subexponential-time algorithm for solving deterministic parity games is given; the running time of that algorithm is $\boldsymbol{n}^{O(\sqrt{\boldsymbol{n}})}$, which improves on [21] when $\boldsymbol{d} \notin O(\sqrt{\boldsymbol{n}})$. No polynomial-time algorithm is known for the parity problem on deterministic games, which lies in NP $\cap$ coNP [17].

Graphs. The parity problem on graphs can be solved in $O(\boldsymbol{m} \log \boldsymbol{d})$ time [24]. Value iteration is not optimal, because the value iteration solutions that have been devised for deterministic games require exponentially many iterations even when applied to the special case of graphs. A reduction from parity objectives to limit-average objectives [20], followed by value iteration for graphs with limit-average objectives [27] (see below), yields an $O(\boldsymbol{m}\boldsymbol{n}^2)$ value iteration solution for the parity problem on graphs, which is still not optimal.

Probabilistic graphs. The parity problem on probabilistic graphs can be solved in polynomial time by first computing the value-1 set T of the given parity objective, and then computing (by linear programming; see Section 3) the probabilistic graph valuation of the reachability objective with target set T [14, 8]. The known value iteration solutions require exponentially many iterations.

Probabilistic games. The parity problem on probabilistic games lies in NP $\cap$ coNP [8]. A value iteration scheme can obtained from the μ-calculus expression for deterministic games, essentially by replacing the deterministic game predecessor operator $\mathsf{maxminPre}$ with the probabilistic game predecessor operator $\mathsf{maxminPre}^P$ [16]. The nested fixpoints are not finitely reachable, but finitely computable, as in the case of Büchi and coBüchi objectives (see Section 4). This leads to a $O(\boldsymbol{\gamma}^{2\boldsymbol{d}})$ value iteration solution for the parity problem on probabilistic games (recall that $\boldsymbol{\gamma}$ is such that $16^{\boldsymbol{n}} \in O(\boldsymbol{\gamma})$).

Concurrent games. The value characterization of [17] for deterministic games can be extended to concurrent games [16]: given a value set with $\boldsymbol{d}$ priorities, the concurrent game valuation $\mathsf{supinf}\,\mathsf{Parity}(p)$ can be defined by a μ-calculus expression of alternation depth $\boldsymbol{d}-1$. The fixpoint expression is very similar to the expression of [17], except that it uses the concurrent game predecessor operator $\mathsf{supinfPre}$ instead of the deterministic game predecessor operator $\mathsf{maxminPre}$. However, as explained in Section 3, no bounds are known for the finite approximability of fixpoints even in the very special case of reachability objectives. The known complexity bounds for the parity problem on concurrent games are the same as for the reachability problem on concurrent games [4] (see Section 3). All fixpoints can be defined in the theory of the reals with addition and multiplication [16]; however, for parity objectives the reduction to the theory of reals yields a decision procedure in 3EXPTIME [16].

5.2 Limit-average objectives

Deterministic games. A value improvement solution for the limit-average problem on deterministic games is given in [35]. For measuring complexity, we consider all values to be rational numbers (rather than reals). Given a reward function $p : S \to \mathbb{Q}_{\geq 0}$, let $\boldsymbol{p} = \sum_{s \in S} |p(s)|$, where $|p(s)|$ denotes the space required to express the rational $p(s)$ in binary. We run Algorithm 1 with the initial valuation $v^0 = p$ and the improvement function

$$\mathsf{Imp}(v)(s) \;=\; \begin{cases} p(s) + \max\{v(s') \mid s' \in E(s)\} & \text{if } \; s \in S_1; \\ p(s) + \min\{v(s') \mid s' \in E(s)\} & \text{if } \; s \in S_2; \end{cases}$$

for all valuations $v \in V$. The value improvement fixpoint is not finitely reachable, but finitely computable. The improvement function is applied $k = \boldsymbol{mn}^3 \cdot 2^{\boldsymbol{p}}$ times. If $v^{i+1} = \mathsf{Imp}(v^i)$ for all $i \geq 0$, then for every state $s \in S$, the value $\sup \mathsf{LimAvg}(p)(s)$ is very close to $v^k(s)/k$. Thus the deterministic game valuation $\sup \mathsf{LimAvg}(p)$ can be constructed from the valuation v^k by applying a suitable limit function [35]. The running time of the value improvement algorithm is pseudo-polynomial. No polynomial-time algorithm is known for the limit-average problem on deterministic games, which lies in NP $\cap$ coNP [35].

Graphs. The limit-average problem on graphs can be solved in $O(\boldsymbol{mn})$ time, by computing the maximum-mean cycle of a directed graph [23]. The best known value iteration solution requires $O(\boldsymbol{mn}^2)$ time [27].

Probabilistic graphs. The limit-average problem on probabilistic graphs can be solved in polynomial time by linear programming [18]. Again value iteration, which requires exponentially many iterations [26, 32], is not optimal.

Probabilistic games. The limit-average problem on probabilistic games lies in NP $\cap$ coNP [25]. No polynomial-time algorithm is known. We conjecture that the value improvement solution of [35], with a suitable modification of the improvement function to account for probabilistic states, can be used to solve the limit-average problem on probabilistic games, and requires exponentially many iterations.

Concurrent games. A value improvement algorithm to compute the values of limit-average objectives for concurrent games is, to our knowledge, not known. The limit-average problem on concurrent games can be solved in EXPTIME [9]. The best known lower bound is PTIME-hardness (by reduction from alternating reachability), which applies already to deterministic games.

5.3 Relation between parity and limit-average objectives

The parity and limit-average problems are related. The parity problem on deterministic games can be polynomial-time reduced to the limit-average problem on deterministic games [20]. The reduction has recently been extended to the case of probabilistic games [6]. The polynomial-time reducibility of concurrent games with parity objectives to concurrent games with limit-average objectives remains an open problem. Since parity objectives lie in $\Sigma_3 \cap \Pi_3$, while limit-average objectives are Π_3-complete, no Wadge reduction [34], and thus no polynomial-time reduction, exists from the limit-average problem to the parity problem on a given class of graphs (e.g., on deterministic games). However, a polynomial-time reduction from deterministic games with limit-average objectives to probabilistic games with reachability objectives is available [35].

6 Concluding Remarks

We briefly mention two topics related to value iteration, which we have not discussed in this paper.

Strategy improvement. An alternative approach to compute the values of games is to iterate over strategies rather than over valuations. Given an objective, a player-1 strategy σ defines a valuation v_σ on graphs and probabilistic graphs, namely, the values that player 1 achieves by following the strategy σ. On deterministic games, probabilistic games, and concurrent games, let v_σ be the optimal valuation obtainable by player 2 if player 1 follows the strategy σ. A strategy improvement algorithm iterates over strategies: given a player-1 strategy σ, the algorithm computes v_σ and then locally improves the strategy σ to achieve a better valuation for player 1. This process is repeated until no improvement is possible. In other words, the strategy improvement approach iterates over local optimizations of strategies instead of over local optimizations of values. For deterministic games with parity objectives, although the best known bound for strategy improvement is exponential, the algorithm works well in practice, and it is an open problem to find a family of examples for which strategy improvement requires a super-polynomial number of iterations [33]. Strategy improvement algorithms are also known for probabilistic games with reachability objectives [12], for probabilistic games with parity objectives [7], and for concurrent games with reachability objectives [5].

Discounted games. An improvement function Imp is *contractive* if there exists a real $0 \leq \beta < 1$ such that for all valuations v_1 and v_2, we have $\mathrm{diff}(\mathsf{Imp}(v_1), \mathsf{Imp}(v_2)) \leq \beta \cdot \mathrm{diff}(v_1, v_2)$. For contractive improvement functions, the value improvement algorithm (Algorithm 1) converges to a fixpoint because of Banach's fixpoint theorem. While the improvement functions we discussed in this paper are not necessarily contractive, in the setting of games with discounted reward objectives [30], improvement functions are contractive. The analysis of discounted games is therefore simpler [15]: (1) a contractive improvement function ensures that there is an unique fixpoint, and hence the nested analysis of least and greatest fixpoints can be avoided; and (2) a contractive improvement function ensures the geometric convergence of valuations, guaranteeing finite approximability even for concurrent games. Moreover, the values of undiscounted games with parity and limit-average objectives can be obtained as appropriate limits of the values of certain discounted reward objectives, as the discount factor goes to 1. This was shown for concurrent games with parity objectives in [15, 19], and for concurrent games with limit-average objectives in [28]. Probabilistic graphs with discounted reward objectives can be solved in polynomial time [18], but no polynomial-time algorithm is known for deterministic games with discounted reward objectives. The problem of computing the values of discounted reward objectives for probabilistic games can be shown to lie in NP $\cap$ coNP.

Acknowledgment. We thank Laurent Doyen for helpful comments on a draft.

References

1. R. Alur, T.A. Henzinger, and O. Kupferman. Alternating-time temporal logic. *Journal of the ACM*, 49:672–713, 2002.
2. A. Chakrabarti, L. de Alfaro, T.A. Henzinger, and M. Stoelinga. Resource interfaces. In *EMSOFT'03*, LNCS 2855, pages 117–133. Springer, 2003.
3. K. Chatterjee. Concurrent games with tail objectives. Technical Report EECS-2005-1385, UC Berkeley, 2005.
4. K. Chatterjee, L. de Alfaro, and T.A. Henzinger. The complexity of quantitative concurrent parity games. In *SODA'06*, pages 678–687. ACM-SIAM, 2006.
5. K. Chatterjee, L. de Alfaro, and T.A. Henzinger. Strategy improvement for concurrent reachability games. In *QEST'06*, pages 291–300. IEEE, 2006.
6. K. Chatterjee and T.A. Henzinger. Reduction of stochastic parity to stochastic mean-payoff games. Technical Report EECS-2006-140, UC Berkeley, 2006.
7. K. Chatterjee and T.A. Henzinger. Strategy improvement and randomized subexponential algorithms for stochastic parity games. In *STACS'06*, LNCS 3884, pages 512–523. Springer, 2006.
8. K. Chatterjee, M. Jurdziński, and T.A. Henzinger. Quantitative stochastic parity games. In *SODA'04*, pages 121–130. ACM-SIAM, 2004.

9. K. Chatterjee, R.Majumdar, and T.A. Henzinger. Stochastic limit-average games are in EXPTIME. Technical Report EECS-2006-143, UC Berkeley, 2006.
10. E.M. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 1999.
11. A. Condon. The complexity of stochastic games. *Information and Computation*, 96:203–224, 1992.
12. A. Condon. On algorithms for simple stochastic games. In *Advances in Computational Complexity Theory*, volume 13 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 51–73. AMS, 1993.
13. C. Courcoubetis and M. Yannakakis. The complexity of probabilistic verification. *Journal of the ACM*, 42:857–907, 1995.
14. L. de Alfaro. *Formal Verification of Probabilistic Systems*. PhD thesis, Stanford University, 1997.
15. L. de Alfaro, T.A. Henzinger, and R. Majumdar. Discounting the future in systems theory. In *ICALP'03*, LNCS 2719, pages 1022–1037. Springer, 2003.
16. L. de Alfaro and R. Majumdar. Quantitative solution of ω-regular games. *Journal of Computer and System Sciences*, 68:374–397, 2004.
17. E.A. Emerson and C. Jutla. Tree automata, μ-calculus, and determinacy. In *FOCS'91*, pages 368–377. IEEE, 1991.
18. J. Filar and K. Vrieze. *Competitive Markov Decision Processes*. Springer, 1997.
19. H. Gimbert and W. Zielonka. Discounting infinite games, but how and why? *ENTCS*, 119:3–9, 2005.
20. M. Jurdziński. Deciding the winner in parity games is in UP $\cap$ coUP. *Information Processing Letters*, 68:119–124, 1998.
21. M. Jurdziński. Small progress measures for solving parity games. In *STACS'00*, LNCS 1770, pages 290–301. Springer, 2000.
22. M. Jurdziński, M. Paterson, and U. Zwick. A deterministic subexponential algorithm for solving parity games. In *SODA'06*, pages 117–123. ACM-SIAM, 2006.
23. R.M. Karp. A characterization of the minimum cycle mean in a digraph. *Discrete Mathematics*, 23:309–311, 1978.
24. V. King, O. Kupferman, and M.Y. Vardi. On the complexity of parity word automata. In *FOSSACS'01*, LNCS 2030, pages 276–286. Springer, 2001.
25. T.A. Liggett and S.A. Lippman. Stochastic games with perfect information and time average payoff. *SIAM Review*, 11:604–607, 1969.
26. M.L. Littman. *Algorithms for Sequential Decision Making*. PhD thesis, Brown University, 1996.
27. O. Madani. Polynomial value iteration algorithms for deterministic MDPs. In *UAI'02*, pages 311–318. Morgan-Kaufmann, 2002.
28. J.F. Mertens and A. Neyman. Stochastic games. *International Journal of Game Theory*, 10:53–66, 1981.
29. S. Safra. On the complexity of ω-automata. In *FOCS'88*, pages 319–327. IEEE, 1988.
30. L.S. Shapley. Stochastic games. *Proceedings of the National Academy of Sciences USA*, 39:1095–1100, 1953.
31. W. Thomas. Languages, automata, and logic. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 3, pages 389–455. Springer, 1997.
32. P. Tseng. Solving H-horizon stationary Markov decision problems in time proportional to log(H). *Operations Research Letters*, 9:287–297, 1990.
33. J. Vöge and M. Jurdziński. A discrete strategy improvement algorithm for solving parity games. In *CAV'00*, pages 202–215. LNCS 1855, Springer, 2000.
34. W.W. Wadge. *Reducibility and Determinateness of Baire Spaces*. PhD thesis, UC Berkeley, 1984.

35. U. Zwick and M. Paterson. The complexity of mean-payoff games on graphs. *Theoretical Computer Science*, 158:343–359, 1996.