Widest Paths and Global Propagation in Bounded Value Iteration for Stochastic Games

Kittiphon Phalakarn, Toru Takisaka, Thomas Haas, Ichiro Hasuo

Solving *stochastic games* with the reachability objective is a fundamental problem, especially in quantitative verification and synthesis. For this purpose, *bounded value iteration (BVI)* attracts attention as an efficient iterative method. However, BVI's performance is often impeded by costly *end component (EC) computation* that is needed to ensure convergence. Our contribution is a novel BVI algorithm that conducts, in addition to local propagation by the Bellman update that is typical of BVI, *global* propagation of upper bounds that is not hindered by ECs. To conduct global propagation in a computationally tractable manner, we construct a weighted graph and solve the *widest path problem* in it. Our experiments show the algorithm's performance advantage over the previous BVI algorithms that rely on EC computation.

1 Introduction

1.1 Stochastic Game (SG)

A *stochastic game* [13] is a two-player game played on a graph. In an SG, an action a of a player causes a transition from the current state s to a successor s', with the latter chosen from a prescribed probability distribution $\delta(s, a, s')$. Under the reachability objective, the two players (called *Maximizer* and *Minimizer*) aim to maximize and minimize, respectively, the reachability probability to a designated target state.

Stochastic games are a fundamental construct in theoretical computer science, especially in the analysis of probabilistic systems. Its complexity is interesting in its own: the problem of threshold reachability—whether Maximizer has a strategy that ensures the reachability probability to be at least given p—is known to be in $\mathsf{UP} \cap \mathsf{coUP}$ [19], but no polynomial algorithm is known. The practical significance of SGs comes from the number of problems that can be encoded to SGs and then solved. Examples include the following: solving deterministic parity games [8], solving stochastic games with the parity or mean-payoff objective [1], and a variety of probabilistic verification and reactive synthesis problems in different application domains such as cyber-physical systems. See e.g. [25].

SGs are often called 2.5-player games, where probabilistic branching is counted as 0.5 players. They generalize deterministic automata (0-player), Markov chains (MCs, 0.5-player), nondeterministic automata (1-player), Markov decision processes (MDPs, 1.5-player) and (deterministic) games (2-

---

* 確率的ゲーム上の有界価値反復法における最大経路幅問題と全域伝播

The content of this paper is previously published in proceedings of the 32nd International Conference on Computer-Aided Verification (CAV 2020) [24].

Kittiphon Phalakarn, ウォータールー大学, University of Waterloo, Waterloo, Canada.

Toru Takisaka, Ichiro Hasuo, 国立情報学研究所, National Institute of Informatics, Tokyo, Japan.

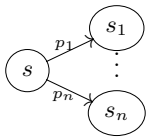
Thomas Haas, ブラウンシュヴァイク工科大学, Technical University of Braunschweig, Braunschweig, Germany.

Ichiro Hasuo, 総合研究大学院大学, The Graduate University for Advanced Studies (SOKENDAI), Tokyo, Japan.

player). Many theoretical considerations on these special cases carry over smoothly to SGs. However, SGs have their peculiarities, too. One example is the treatment of end components in bounded value iteration, as we describe later.

1.2 Value Iteration (VI)

In an SG, we are interested in the *optimal* reachability probability, that is, the reachability probability when both Maximizer and Minimizer take their optimal strategies. The function that returns these optimal reachability probabilities is called the *value function* $V(\mathcal{G})$ of the SG $\mathcal{G}$; our interest is in computing this value function, desirably constructing optimal strategies for the two players at the same time. For this purpose, two principal families of solution methods are *strategy iteration (SI)* [19] and *value iteration (VI)* [10,13]—the latter is commonly preferred for performance reasons.



The mathematical principle that underpins VI is the characterization of the value function $V(\mathcal{G})$ as the *least fixed point (lfp)* of an function update operator $\mathbb{X}$ called the *Bellman operator*. The Bellman operator $\mathbb{X}$ back-propagates function values by one step, using the average. For the simple case of Markov chains shown on the right, it is defined by $(\mathbb{X}f)(s) = \sum_i p_i \cdot f(s_i)$, turning a function $f\colon S \to [0,1]$ (i.e., assignment of "scores" to states) to $\mathbb{X}f\colon S \to [0,1]$.

Since $V(\mathcal{G})$ is the lfp $\mu\mathbb{X}$, Kleene's fixed point theorem tells us the sequence

$$\bot \;\leq\; \mathbb{X}\bot \;\leq\; \mathbb{X}^2\bot \;\leq\; \cdots, \tag{1}$$

where $\bot$ is the least element of the function space $S \to [0,1]$, converges to $V(\mathcal{G}) = \mu\mathbb{X}$. VI consists of the iterative approximation of $V(\mathcal{G})$ via the sequence (1).

An issue from the practical point of view, however, is that $\mathbb{X}^i\bot$ never becomes equal to $V(\mathcal{G})$ in general. Even worse, one cannot know how close the current approximant $\mathbb{X}^i\bot$ is to the desired function $V(\mathcal{G})$ [18]. In summary, VI as an iterative approximation method does not give any precision guarantee.

1.3 Bounded Value Iteration (BVI) and End Components

Bounded value iteration (BVI) has been actively studied as an extension of VI that comes with a precision guarantee [2, 3, 5, 16, 18, 20, 23]. Its core ideas are the following two.

Firstly, BVI computes not only iterative lower bounds $L_i = \mathbb{X}^i\bot$ for $V(\mathcal{G})$, but also iterative *upper bounds* U_i, as shown on the right in (2). This gives us a precision guarantee—$V(\mathcal{G})$ must lie between the approximants L_i and U_i.

$$\begin{array}{l} L_0 \leq L_1 \leq \cdots \\ U_0 \geq U_1 \geq \cdots \end{array} \searrow\!\nearrow\; V(\mathcal{G}) \tag{2}$$

Secondly, for computing upper bounds U_i, BVI uses the Bellman operator again: $U_i = \mathbb{X}^i\top$ where $\top$ is the greatest element of the function space $S \to [0,1]$. This leads to the following approximation sequence that is dual to (1):

$$\top \;\geq\; \mathbb{X}\top \;\geq\; \mathbb{X}^2\top \;\geq\; \cdots. \tag{3}$$

The sequence (3) converges to the *greatest fixed point (gfp)* $\nu\mathbb{X}$ of $\mathbb{X}$, which must be above the lfp $V(\mathcal{G}) = \mu\mathbb{X}$. Therefore the elements in (3) are all above $V(\mathcal{G})$.

The problem, however, is that the gfp $\nu\mathbb{X}$ is not necessarily the same as $\mu\mathbb{X}$. Therefore the upper bounds $U_0 \geq U_1 \geq \cdots$ given by (3) may not converge to $V(\mathcal{G})$. In other words, for a given threshold $\varepsilon > 0$, the bounds in (2) may fail to achieve $U_i - L_i \leq \varepsilon$, no matter how large i is.

In the literature, the source of this convergence issue has been identified to

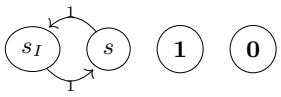


図 1: A Markov chain (MC) for which the naive BVI fails to converge

be *end components (ECs)* in MCs/MDPs/SGs. ECs are much like loops without exits—an example is in Fig. 1, where we use a Markov chain (MC) for simplicity. Any function f that assigns the same value to the states s_I and s can be a fixed point of the Bellman operator $\mathbb{X}$ (that back-propagates f by averages); therefore, the gfp $\nu\mathbb{X}$ assigns 1 to both s_I and s. In contrast, $(\mu\mathbb{X})(s_I) = (\mu\mathbb{X})(s) = 0$, which says one never reaches the target $\mathbf{1}$ from s_I or s (which is obvious).

Most previous works on BVI have focused on the problem of how to deal with ECs. Their solutions are to get somehow rid of ECs. For example, ECs in MDPs are discovered and *collapsed* in [5, 18]; ECs in SGs cannot simply be collapsed, and an elaborate method is proposed in the recent work [20] that *deflates* them. This is the context of the current work, and we aim to enhance BVI for SGs.

1.4 Contribution: Global Propagation in BVI with Widest Paths

The algorithms in [20] seem to be the only BVI algorithms known for SGs. In their performance, however, EC computation often becomes a bottleneck. Our contribution in this paper is a new BVI algorithm for SGs that is *free from the need for EC computation*.

The key idea of our algorithm is *global propagation* for upper bounds, as sketched below. In each iteration for upper bounds $U_0 \geq U_1 \geq \cdots$, we conduct *global* propagation, in addition to the *local* propagation in the usual BVI. The latter means the application of $\mathbb{X}$ to $\mathbb{X}^i\top$, leading to $\mathbb{X}^{i+1}\top$; this local propagation, as we previously discussed, gets trapped in end components. In contrast, our global propagation looks at paths from each state s to the target $\mathbf{1}$, ignoring end components. For example, in Fig. 1, our global propagation sees that there is no path from s_I to the target $\mathbf{1}$, and consequently assigns 0 as an upper bound for the value function $V(\mathcal{G})(s_I)$.

Such global propagation is easier said than done—in fact, the very advantage of VI is that the *global* quantities (namely reachability probabilities) get computed by iterations of *local* propagation. Conducting global propagation in a computationally tractable manner requires a careful choice of its venue. The solution in this paper is to compute *widest paths* in a suitable (directed) weighted graph.

More specifically, in each iteration where we compute an upper bound U_i, we conduct the following operations.

- (Player reduction) We turn the given SG $\mathcal{G}$ into an MDP $\mathcal{M}_i$, by restricting Minimizer's actions to the *empirically optimal* ones. The latter means they are optimal with respect to the current under-approximation L_i of $V(\mathcal{G})$.
- (Local propagation) The MDP $\mathcal{M}_i$ is then turned into a weighted graph (WG) $\mathcal{W}_i$. The construction of $\mathcal{W}_i$ consists of the application of $\mathbb{X}$ to the previous bound U_{i-1} (i.e. local propagation), and forgetting the information that cannot be expressed in a weighted graph (such as the precise transition probabilities $\delta(s, a, s')$ that depend on the action a).
  Due to this information loss, our analysis in $\mathcal{W}_i$ is necessarily approximate. Nevertheless, the benefit of $\mathcal{W}_i$'s simplicity is significant, as in the following step.
- (Global propagation) In the WG $\mathcal{W}_i$, we solve the *widest path problem*. This classic graph-theoretic problem can be solved efficiently, e.g., by the Dijkstra algorithm. The widest path width gives a new upper bound U_i.

Our algorithm satisfies soundness ($V(\mathcal{G}) \leq U_i$) and convergence ($U_i \to V(\mathcal{G})$ as $i \to \infty$). That the upper bounds decrease ($U_0 \geq U_1 \geq \cdots$) will be obvious by construction. Hence, it always terminates and returns an approximation of optimal

reachability probability within the required precision. Proofs of these properties can be found in [24].

We have also implemented our algorithm. Our experiments compare its performance to the algorithms from [20] (the original one and its learning-based variation). The results show our consistent performance advantage: depending on SGs, our performance is from comparable to dozens of times faster. The advantage is especially eminent in SGs with many ECs.

1.5 Related Works

VI and BVI have been pursued principally for MDPs. The only work we know that deals with SGs is [20]—with the exception of [26] that works in a restricted setting where every end component belongs exclusively to either player. The work closest to ours is therefore [20], in that we solve the same problem.

For MDPs, the idea of BVI is first introduced in [23]; they worked in a limited setting where ECs do not cause the convergence issue. Its extension to general MDPs with the reachability objective is presented in [5, 18], where ECs are computed and then collapsed. BVI is studied under different names in these works: *bounded real time dynamic programming* [5,23] and *interval iteration* [18]. The work [20] is an extension of this line of work from MDPs to SGs.

The work [20] has seen a few extensions to more advanced settings: black-box settings [3], concurrent reachability [16], and generalized reachability games [2].

Most BVI algorithms involve EC computation (although ours does not). The EC algorithm in [14,15] is used in [18,20]; more recent algorithms include [7,9].

1.6 Organization

In §2 we present some preliminaries. In §3 we review VI and BVI with an emphasis on the role of Kleene's fixed point theorem. This paves the way to §4 where we present our algorithm, as well as its soundness and convergence properties. Experiment results are shown in §5.

2 Preliminaries

We fix some basic notations. Let X be a set. We let X^* denote the set of finite sequences over X, that is, $X^* = \bigcup_{i\in\mathbb{N}} X^i$. We let $X^+ = X^* \setminus \{\varepsilon\}$, where ε denotes the empty sequence (of length 0). The set of infinite sequences over X is denoted by X^ω. The set of functions from X to Y is denoted by $X \to Y$.

2.1 Stochastic Games

In a stochastic game, two players (*Maximizer* $\Box$ and *Minimizer* $\bigcirc$) play against each other. The goals of the two players are to maximize and minimize the *value function*, respectively. Many different definitions are possible for value functions. In this paper (as well as all the works on (bounded) value iteration), we focus on the *reachability objective*, in which case a value function is defined by the reachability probability to a designated target state $\mathbf{1}$.

Definition 2.1 (stochastic game) A stochastic game (SG) is a tuple $\mathcal{G} = (S, S_\Box, S_\bigcirc, s_I, \mathbf{1}, \mathbf{0}, A, \mathrm{Av}, \delta)$ where

- S is a finite set of *states*, partitioned into $S_\Box$ and $S_\bigcirc$ (i.e., $S = S_\Box \cup S_\bigcirc$, $S_\Box \cap S_\bigcirc = \emptyset$). $s \in S_\Box$ is *Maximizer's* state; $s \in S_\bigcirc$ is *Minimizer's* state.
- $s_I \in S$ is an *initial* state, $\mathbf{1} \in S_\Box$ is a *target*, and $\mathbf{0} \in S_\bigcirc$ is a sink.
- A is a finite set of *actions*.
- $\mathrm{Av} : S \to 2^A$ defines the set of actions that are

available at each state $s \in S$.

- $\delta : S \times A \times S \to [0,1]$ is a *transition function*, where $\delta(s,a,s')$ gives a probability with which to reach the state s' when the action a is taken at the state s. The value $\delta(s,a,s')$ is non-zero only if $a \in \mathrm{Av}(s)$; it must satisfy $\sum_{s' \in S} \delta(s,a,s') = 1$ for all $s \in S$ and $a \in \mathrm{Av}(s)$.

We assume that each of **1** and **0** allows only one action that leads to a self-loop with probability 1. Moreover, for theoretical convenience, we assume that all SGs are non-blocking. That is, $\mathrm{Av}(s) \neq \emptyset$ for each $s \in S$.

We introduce some notations: $\mathrm{post}(s,a) = \{s' \mid \delta(s,a,s') > 0\}$, and for $S' \subseteq S$, we let $S'_\Box = S' \cap S_\Box$ and $S'_\bigcirc = S' \cap S_\bigcirc$.

Definition 2.2 (MDP, MC) An SG such that $S_\Box = S \setminus \{\mathbf{0}\}$ (i.e. Minimizer is absent) is called a *Markov decision process (MDP)*. We often omit the second and third components for MDPs, writing $\mathcal{M} = (S, s_I, \mathbf{1}, \mathbf{0}, A, \mathrm{Av}, \delta)$.

An SG such that $|\mathrm{Av}(s)| = 1$ for each $s \in S$—both Maximizer and Minimizer are absent—is called a *Markov chain (MC)*. It is also denoted simply by a tuple $\mathcal{G} = (S, s_I, \mathbf{1}, \mathbf{0}, \delta)$ where its transition function is of the type $\delta : S \times S \to [0,1]$.

Every notion for SGs that appears below applies to MDPs and MCs, too.

Example 2.3 Fig. 2 presents an example of an SG. At the state s_1 of Minimizer, two actions α and β are in $\mathrm{Av}(s_1)$. If Minimizer chooses α, the next state is s_2 with probability $\delta(s_1, \alpha, s_2) = 1$. If Minimizer instead chooses β, the next state is **1** with probability $\delta(s_1, \beta, \mathbf{1}) = 0.8$ or **0** with probability $\delta(s_1, \beta, \mathbf{0}) = 0.2$.

Maximizer's goal is to reach **1** as often as possible by choosing suitable actions. Minimizer's goal is to avoid reaching **1**—this can be achieved, for example but not exclusively, by reaching **0**.

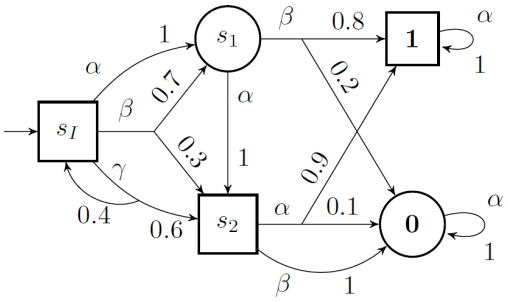


図 2: A stochastic game (SG), an example

Both players choose their actions according to their *strategies*. It is well-known [13] that *positional* (also called *memoryless*) and *deterministic* (also called *pure*) strategies are complete for finite SGs with the reachability objective.

Definition 2.4 (strategy, path) Let $\mathcal{G}$ be the SG in Def. 2.1. A *strategy* for Maximizer in $\mathcal{G}$ is a function $\sigma \colon S_\Box \to A$ such that $\sigma(s) \in \mathrm{Av}(s)$ for each $s \in S_\Box$. A *strategy* for Minimizer is defined similarly. The set of Maximizer's strategies in $\mathcal{G}$ is denoted by $\mathrm{str}^{\mathcal{G}}_\Box$; that of Minimizer's is denoted by $\mathrm{str}^{\mathcal{G}}_\bigcirc$.

Strategies $\tau \in \mathrm{str}^{\mathcal{G}}_\Box$ and $\sigma \in \mathrm{str}^{\mathcal{G}}_\bigcirc$ in $\mathcal{G}$ turn the game $\mathcal{G}$ into a Markov chain, which is denoted by $\mathcal{G}^{\tau,\sigma}$. Similarly, a strategy τ for Maximizer (who is the only player) in an MDP $\mathcal{M}$ induces an MC, denoted by $\mathcal{M}^\tau$.

A sequence $s_0 a_0 s_1 a_1 s_2 a_2 \ldots \in (S \times A)^\omega$ is called an *infinite path* in $\mathcal{G}$ if for all $i \in \mathbb{N}$ it satisfies $a_i \in \mathrm{Av}(s_i)$ and $s_{i+1} \in \mathrm{post}(s_i, a_i)$. A prefix $s_0 a_0 s_1 \ldots s_k$ of an infinite path ending with a state is called a *finite path*. If $\mathcal{G}$ is an MC, then we omit actions in a path and write $s_0 s_1 s_2 \ldots$ or $s_0 s_1 \ldots s_k$.

Given a game $\mathcal{G}$ and strategies τ, σ for the two players, the induced MC $\mathcal{G}^{\tau,\sigma}$ assigns to each state $s \in S$ a probability distribution $\mathbb{P}^{\tau,\sigma}_s$. The distribution is with respect to the standard measurable

structure of S^ω; see, e.g., [4, Chap. 10]. For each measurable subset $X \subseteq S^\omega$, $\mathbb{P}_s^{\tau,\sigma}(X)$ is the probability with which $\mathcal{G}^{\tau,\sigma}$, starting from the state s, produces an infinite path π that belongs to X.

It is well-known that all the LTL properties are measurable in S^ω. In the current setting with the reachability objective, we are interested in the probability of eventually reaching $\mathbf{1}$, denoted by $\mathbb{P}_s^{\tau,\sigma}(\Diamond\mathbf{1})$.

Definition 2.5 (value function $V(\mathcal{G})$) Let $\mathcal{G}$ be the SG in Def. 2.1. The *value function* $V(\mathcal{G})$ of $\mathcal{G}$ is defined by

$$\begin{aligned} V(\mathcal{G})(s) &= \max_{\tau \in \mathrm{str}^{\mathcal{G}}_{\Box}} \min_{\sigma \in \mathrm{str}^{\mathcal{G}}_{\bigcirc}} \mathbb{P}_s^{\tau,\sigma}(\Diamond\mathbf{1}) \\ &= \min_{\sigma \in \mathrm{str}^{\mathcal{G}}_{\bigcirc}} \max_{\tau \in \mathrm{str}^{\mathcal{G}}_{\Box}} \mathbb{P}_s^{\tau,\sigma}(\Diamond\mathbf{1}), \end{aligned}$$

where the last equality is shown in [13].

We say a strategy τ of Maximizer's is *optimal* if $V(\mathcal{G})(s) = \min_\sigma \mathbb{P}_s^{\tau,\sigma}(\Diamond\mathbf{1})$ for each $s \in S$; similarly, we say a strategy σ of Minimizer's is *optimal* if $V(\mathcal{G})(s) = \max_\sigma \mathbb{P}_s^{\sigma,\tau}(\Diamond\mathbf{1})$ for each $s \in S$.

We write V for $V(\mathcal{G})$ when the dependence on $\mathcal{G}$ is clear from the context.

The set of states with a non-zero value is denoted by $S_{\Diamond\mathbf{1}}$. That is, $S_{\Diamond\mathbf{1}} = \{s \in S \mid V(\mathcal{G})(s) > 0\}$.

Example 2.6 Consider the SG $\mathcal{G}$ from Fig. 2. At s_2, Maximizer's action should be α. Hence, $V(\mathcal{G})(s_2) = 0.9$. At s_1, if Minimizer chooses α, then the probability of reaching $\mathbf{1}$ will be 0.9 by $V(\mathcal{G})(s_2)$. Thus, Minimizer should choose β at s_1, which yields $V(\mathcal{G})(s_1) = 0.8$. Finally, at s_I, γ is the best choice, since Maximizer can choose this action infinitely often until it gets to s_2. We have $V(\mathcal{G})(s_I) = 0.9$.

2. 2 The Widest Path Problem

Definition 2.7 (weighted graph) A (directed) *weighted graph (WG)* is a triple $\mathcal{W} = (V, E, w)$ of a finite set V of *vertices*, a set $E \subseteq V \times V$ of *edges*, and a *weight function* $w\colon E \to [0, 1]$ where $[0, 1]$ is the unit interval.

A (finite) *path* in a WG is defined in the usual graph-theoretic way.

In the widest path problem, an edge weight $w(v, v')$ is thought of as its capacity, and the capacity of a path is determined by its bottleneck. The problem asks for a path with the greatest capacity. In this paper, we use the following *all-source single-destination* version of the problem.

Definition 2.8 (the widest path problem) A (finite) *path* in $\mathcal{W} = (V, E, w)$ is a sequence $v_0 v_1 \dots v_n$ of vertices such that $(v_i, v_{i+1}) \in E$ for each $i \in [0, n-1]$. The *width* of a path $v_0 v_1 \dots v_n$ is given by $\min_{i \in [0,n-1]} w(v_i, v_{i+1})$.

The *widest path problem (WPP)* is the following problem.

Given: a WG $\mathcal{W} = (V, E, w)$ and a target vertex $v_\mathrm{t} \in V$.

Answer: for each $v \in V$, the widest width of the paths from v to v_t, that is,

$$\max_{n \in \mathbb{N}, v = v_0, v_1, \dots, v_n = v_\mathrm{t}} \min_{i \in [0,n-1]} w(v_i, v_{i+1}),$$

We let $\mathrm{WPW}(\mathcal{W}, v_\mathrm{t})$ denote a function that solves this problem, and let $\mathrm{WPath}(\mathcal{W}, v_\mathrm{t})$ denote a function that assigns to each $v \in V$ a widest path to v_t. Furthermore, we assume the following property of WPath: if $\mathrm{WPath}(\mathcal{W}, v_\mathrm{t})(v_0) = v_0 v_1 \dots v_k v_\mathrm{t}$, then $\mathrm{WPath}(\mathcal{W}, v_\mathrm{t})(v_i) = v_i v_{i+1} \dots v_k v_\mathrm{t}$ for each $i \in [0, k]$.

Efficient algorithms are known for $\mathrm{WPW}(\mathcal{W}, v_\mathrm{t})$. An example is the Dijkstra search algorithm with Fibonacci heaps [17]; it is originally for the single-source all-destination version but its adaptation is easy. The algorithm runs in time $O(|E| + |V| \log |V|)$. It returns a widest path in addition to its width, too, computing the function $\mathrm{WPath}(\mathcal{W}, v_\mathrm{t})$ with the property required in the

above.

3 (Bounded) Value Iteration

3.1 Bellman Operator and Value Iteration

The following construct—used for "local propagation" in computing the value function—is central to formal analysis of probabilistic systems and games.

Definition 3.1 (Bellman Operator) Let $\mathcal{G} = (S, S_\Box, S_\bigcirc, s_I, \mathbf{1}, \mathbf{0}, A, \mathrm{Av}, \delta)$ be a stochastic game. For each state $s \in S$, an available action $a \in \mathrm{Av}(s)$, and $f : S \to [0,1]$, we define a function $\mathbb{X}_a f : S \to [0,1]$ by the following.

$$(\mathbb{X}_a f)(s) = \begin{cases} 1 & \text{if } s = \mathbf{1}, \\ 0 & \text{if } s = \mathbf{0}, \\ \sum_{s' \in S} \delta(s,a,s') \cdot f(s') & \text{if } s \neq \mathbf{0}, \mathbf{1}. \end{cases}$$

These functions are used in the following definition of the *Bellman operator* $\mathbb{X}\colon (S \to [0,1]) \to (S \to [0,1])$ over $\mathcal{G}$:

$$(\mathbb{X} f)(s) = \begin{cases} \max_{a \in \mathrm{Av}(s)} (\mathbb{X}_a f)(s) & \text{if } s \in S_\Box \\ \min_{a \in \mathrm{Av}(s)} (\mathbb{X}_a f)(s) & \text{if } s \in S_\bigcirc \end{cases}$$

The function space $S \to [0,1]$ inherits the usual order $\leq$ between real numbers in the unit interval $[0,1]$, that is, $f \leq g$ if $f(s) \leq g(s)$ for each $s \in S$. The Bellman operator $\mathbb{X}$ over $S \to [0,1]$ is clearly monotone; it is easily seen to preserve max and min, using the fact that the state space S of an SG is finite. Therefore we obtain the following, as consequences of Kleene's fixed point theorem.

Lemma 3.2 *Assume the setting of Def. 3.1.*

1. *The Bellman operator $\mathbb{X}$ has the greatest fixed point (gfp) $\nu\mathbb{X}\colon S \to [0,1]$. It is obtained as the limit of the descending ω-chain*
$$\top \geq \mathbb{X}\top \geq \mathbb{X}^2\top \geq \cdots,$$
where $\top$ is the greatest element of $S \to [0,1]$ (i.e., $\top(s) = 1$ for each $s \in S$). In other words, we have
$$(\nu\mathbb{X})(s) = \inf_{i \in \mathbb{N}} \big((\mathbb{X}^i \top)(s)\big)$$
for each $s \in S$.
2. *Symmetrically, $\mathbb{X}$ has the least fixed point (lfp) $\mu\mathbb{X}\colon S \to [0,1]$, obtained as the limit of the ascending chain*
$$\bot \leq \mathbb{X}\bot \leq \mathbb{X}^2\bot \leq \cdots, \qquad (4)$$
where $\bot(s) = 0$ for each $s \in S$. That is, we have
$$(\mu\mathbb{X})(s) = \sup_{i \in \mathbb{N}} \big((\mathbb{X}^i \bot)(s)\big)$$
for each $s \in S$.

The following characterization is fundamental. See, e.g., [10].

Theorem 3.3 *Let $\mathcal{G}$ be a stochastic game. The value function $V = V(\mathcal{G})$ (Def. 2.5) coincides with the least fixed point $\mu\mathbb{X}$.*

The fact that $V(\mathcal{G})$ is the least fixed point of $\mathbb{X}$ implies the following: a strategy τ of Maximizer is optimal if and only if $\big(\mathbb{X}_{\tau(s)}\big(V(\mathcal{G})\big)\big)(s) = V(\mathcal{G})(s)$ holds for each $s \in S_\Box$; similarly for Minimizer. We say $a \in \mathrm{Av}(s)$ is *optimal* at s if $\mathbb{X}_a V(\mathcal{G})(s) = V(\mathcal{G})(s)$ holds; otherwise a is *suboptimal*.

Lem. 3.2.2 & Thm. 3.3 suggest iterative *under*-approximation of $V(\mathcal{G})$ by $\bot \leq \mathbb{X}\bot \leq \mathbb{X}^2\bot \leq \cdots$. This is the principle of *value iteration* (VI); see Algorithm 1.

Example 3.4 The values L_i computed by Algorithm 1, for the SG in Fig. 2, are shown in the following table. The values at $\mathbf{0}$ and $\mathbf{1}$ are omitted.

s	L_0	L_1	L_2	L_3	L_4	L_5	...	$V(\mathcal{G})$
s_I	0	0	0.54	0.83	0.87	0.89		0.9
s_1	0	0	0.8	0.8	0.8	0.8	...	0.8
s_2	0	0.9	0.9	0.9	0.9	0.9		0.9

$L_i(s_I)$ converges to, but is never equal to, $V(\mathcal{G})(s_I)$. The converges rate can be arbitrarily slow: for any $\varepsilon \in (0,1)$ and $k \in \mathbb{N}$ there is an SG $\mathcal{G}$ and a state s such that $V(\mathcal{G})(s) - L_k(s) > \varepsilon$. One

Algorithm 1: Value iteration (VI) for a stochastic game $\mathcal{G} = (S, S_\Box, S_\bigcirc, s_I, \mathbf{1}, \mathbf{0}, A, \mathrm{Av}, \delta)$ and a stopping threshold $\Delta > 0$

```
1 procedure VI(𝒢, Δ)
2     L_0 ← ⊥                                          // Initialize lower bound
3     while L_i(s_I) − L_{i−1}(s_I) < Δ do             // Typical stopping criterion
4         i++
5         L_i ← 𝕏L_{i−1}                               // Bellman update
6     return L_i(s_I)
```

sees this by modifying Fig. 2 with $\delta(s_I, \gamma, s_2) = \varepsilon'$ and $\delta(s_I, \gamma, s_I) = 1-\varepsilon'$, where $\varepsilon' > 0$ is an arbitrary small positive constant.

There is no known stopping criterion for VI (Algorithm 1) with a precision guarantee, besides the one in [10] that is too pessimistic to be practical. The one shown in Line 3 ("little progress") is a commonly used heuristic, but it is known to lead to arbitrarily wrong results [18].

3.2 Bounded Value Iteration

When we turn back to Lem. 3.2, Lem. 3.2.1 suggests another iterative approximation, namely *over*-approximation of the value function V by $\top \geq \mathbb{X}\top \geq \mathbb{X}^2\top \geq \cdots$. The chain converges to the gfp $\nu\mathbb{X}$ that is necessarily above the lfp $\mu\mathbb{X}$. This is the principle that underlies *bounded value iteration* (BVI); see Algorithm 2 for its naive prototype. BVI has been actively studied in the literature [2, 3, 5, 16, 18, 20, 23], sometimes under different names (such as *bounded real time dynamic programming* [5, 23] or *interval iteration* [18]).

BVI comes with a precision guarantee: since $V(\mathcal{G})$ lies between L_i and U_i (whose gap is at most ε), the approximation L_i is at most ε apart from $V(\mathcal{G})$.

The catch, however, is that $\mu\mathbb{X}$ and $\nu\mathbb{X}$ may not coincide, and therefore the overapproximation might not converge to the desired $\mu\mathbb{X}$. This means Algorithm 2 might not terminate. This is the main technical challenge addressed in the previous works on BVI, including [5, 20].

In those works, the source of the failure of convergence is identified to be *end components*. See the (very simple) Markov chain in Fig. 1, where the reachability probability from s_I to $\mathbf{1}$ is clearly 0. However, due to the loop between s_I and s, the values $U_i(s_I)$ and $U_i(s)$—these get updated to the average of U_{i-1} at successors—are easily seen to remain 1. Roughly speaking, end components generalize such loops defined in MDPs and SGs (the definitions are graph-theoretic, in terms of strongly connected components). End components cause non-convergence of naive BVI, essentially for the reason we just described.

The solutions previously proposed to this challenge have been to "get rid of end components." For MDPs (1.5 players), the *collapsing* technique detects end components and collapses each of them into a single state [5, 18]. After doing so, the Bellman operator $\mathbb{X}$ has a unique fixed point (therefore $\mu\mathbb{X} = \nu\mathbb{X}$), assuring convergence of BVI (Algorithm 2). In the case of SGs (2.5 players), end components cannot simply be collapsed into single states—they must be handled carefully, taking the "best exits" into account. This is the key idea of the *deflating* technique proposed for SGs in [20].

4 Our Algorithm: Bounded Value Iteration with Upper Bounds Given by

Algorithm 2: Bounded value iteration (BVI) for a stochastic game $\mathcal{G} = (S, S_\Box, S_\bigcirc, s_I, \mathbf{1}, \mathbf{0}, A, \mathrm{Av}, \delta)$ and a stopping threshold $\varepsilon > 0$—a naive prototype that suffers from end components

```
1 procedure VI(𝒢, ε)
2     L_0 ← ⊥, U_0 ← ⊤                          // Initialize lower and upper bound
3     while U_i(s_I) − L_i(s_I) > ε do          // Check the gap at the initial state
4         i++
5         L_i ← 𝕏L_{i−1}, U_i ← 𝕏U_{i−1}        // Bellman update
6     return L_i(s_I)
```

Widest Paths

In our algorithm, like in other BVI algorithm, we iteratively construct upper and lower bounds U_i, L_i of the value function $V(\mathcal{G})$ at the same time. See (2). In updating U_i, however, we go beyond the *local* propagation by the Bellman update and conduct *global* propagation, too. This frees us from the curse of end components. The outline of our algorithm is as follows.

- The lower bound L_i is given by $L_i = \mathbb{X}^i \bot$, following Lem. 3.2.2 and Thm. 3.3. This is the same as the other VI algorithms.
- The upper bounds U_i is constructed in the following three steps, using a *global* propagation that takes advantage of fast widest path algorithms.
  - **(Player reduction)** Firstly, we turn the SG $\mathcal{G}$ into an MDP $\mathcal{M}_i$ by fixing Minimizer's strategy to a specific one σ_i.
    Any choice of σ_i would do for the sake of *soundness* (that is, $V(\mathcal{G}) \leq U_i$). However, for *convergence* (that is, $U_i \to V(\mathcal{G})$ as $i \to \infty$), it is important to have $\sigma_0, \sigma_1, \ldots$ eventually converge to Minimizer's optimal strategy $\sigma_\bigcirc$. Therefore we let L_i—the current lower estimate of $V(\mathcal{G})$—induce σ_i. Recall that L_i converges to $V(\mathcal{G})$ (Lem. 3.2.2, Thm. 3.3).
  - **(Preprocessing by local propagation)** Secondly, we turn the MDP $\mathcal{M}_i$ into a weighted graph (WG) W_i.
    The construction here is *local* propagation of the previous upper bound U_{i-1}, from each state s to its predecessors in $\mathcal{M}_i$. This is much like an application of the Bellman operator $\mathbb{X}$.
  - **(Global propagation by widest paths)** Finally, we solve the widest path problem in the WG W_i, from each state s to the target state $\mathbf{1}$. The maximum path width from s to $\mathbf{1}$ is used as the value of the upper bound $U_i(s)$.
    This way, we conduct *global* propagation of upper bounds, for which end components pose no threats. Our global propagation is still computationally feasible, thanks to the preprocessing in the previous step that turns a problem on an MDP into one on a WG (modulo some sound approximation).

The use of *global* propagation for upper bounds is a distinguishing feature of our algorithm. This is unlike other BVI algorithms (such as [5, 20]) where upper-bound propagation is only local and stepwise. The latter gets trapped when it encounters an EC—therefore some trick such as collapsing [5] and deflating [20] is needed—while our global propagation looks directly at the target state $\mathbf{1}$.

The above outline is presented as pseudocode in Algorithm 3. We describe the three steps in the rest of the section. In particular, we exhibit the

Algorithm 3: Our BVI algorithm via widest paths. Here $\mathcal{G} = (S, S_\Box, S_\bigcirc, s_I, \mathbf{1}, \mathbf{0}, A, \mathrm{Av}, \delta)$ is an SG; $\varepsilon > 0$ is a stopping threshold.

1 **procedure** BVI_WP($\mathcal{G}, \varepsilon$)
2 $L_0 \leftarrow \bot$, $U_0 \leftarrow \top$, $i \leftarrow 0$
3 **while** $U_i(s_I) - L_i(s_I) > \varepsilon$ **do**
4 i++
5 $L_i \leftarrow \mathbb{X} L_{i-1}$ `// value iteration for lower bounds`
6 $\mathcal{M}_i \leftarrow \mathcal{M}_{\mathrm{PlRd}}(\mathcal{G}, L_i)$ `// player reduction`
7 $\mathcal{W}_i \leftarrow \mathcal{W}_{\mathrm{LcPg}}(\mathcal{M}_i, U_{i-1})$ `// local propagation`
8 $U_i \leftarrow \min\{U_{i-1}, \mathrm{WPW}(\mathcal{W}_i)\}$ `// widest path computation`
9 **return** $U_i(s_I)$

definitions of $\mathcal{M}_{\mathrm{PlRd}}$ and $\mathcal{W}_{\mathrm{LcPg}}$ (WPW has been defined and discussed in Def. 2.8) as well as correctness of the algorithm (§4. 3). Proofs of theorems in this section can be found in [24].

4. 1 Player Reduction: from SGs to MDPs

In Line 6, Algorithm 3 reduces the input SG $\mathcal{G}$ into an MDP $\mathcal{M}_1$ by 1) restricting actions (from Av to Av′), and 2) turning Minimizer's states into Maximizer's.

Definition 4.1 (the MDP $\mathcal{M}(\mathcal{G}, \mathrm{Av}')$) Let $\mathcal{G}$ be the game in Algorithm 3, and $\mathrm{Av}' \colon S \to 2^A$ be such that $\emptyset \neq \mathrm{Av}'(s) \subseteq \mathrm{Av}(s)$ for each $s \in S$.

Then the MDP given by the tuple $(S, S \setminus \{\mathbf{0}\}, \{\mathbf{0}\}, s_I, \mathbf{1}, \mathbf{0}, A, \mathrm{Av}', \delta)$ shall be denoted by $\mathcal{M}(\mathcal{G}, \mathrm{Av}')$, and we say it is induced from $\mathcal{G}$ by restricting Av to Av′.

The following class of action restrictions will be heavily used.

Definition 4.2 (Minimizer restriction) Let $\mathcal{G}$ be as in Algorithm 3. A *Minimizer restriction* of Av is a function $\mathrm{Av}' \colon S \to 2^A$ such that 1) $\emptyset \neq \mathrm{Av}'(s) \subseteq \mathrm{Av}(s)$ for each $s \in S$, and 2) $\mathrm{Av}'(s) = \mathrm{Av}(s)$ for each state $s \in S_\Box$ of Maximizer's.

In Algorithm 3, we will be using the MDP induced by the following specific Minimizer restriction induced by a function f.

Definition 4.3 (the MDP $\mathcal{M}_{\mathrm{PlRd}}(\mathcal{G}, f)$) Let $\mathcal{G}$ be the game in Algorithm 3, and $f \colon S \to [0, 1]$ be a function. The MDP $\mathcal{M}_{\mathrm{PlRd}}(\mathcal{G}, f)$ is defined to be $\mathcal{M}(\mathcal{G}, \mathrm{Av}_f)$ (Def. 4.1), where the function $\mathrm{Av}_f \colon S \to 2^A$ is defined as follows: for $s \in S_\Box$ we let $\mathrm{Av}_f(s) = \mathrm{Av}(s)$; and for $s \in S_\bigcirc$ we let

$$\mathrm{Av}_f(s) = \{a \in \mathrm{Av}(s) \mid \forall b \in \mathrm{Av}(s).\, (\mathbb{X}_a f)(s) \leq (\mathbb{X}_b f)(s)\}. \tag{5}$$

The function Av_f is a Minimizer restriction in $\mathcal{G}$ (Def. 4.2).

The intuition of (5) is that $a = \arg\min_{b \in \mathrm{Av}(s)} (\mathbb{X}_b f)(s)$. In the use of this construction in Algorithm 3, the function f will be our "best guess" L_i of the value function $V(\mathcal{G})$. In this situation, $\arg\min_{b \in \mathrm{Av}(s)} (\mathbb{X}_b f)(s)$ is the best action for Minimizer based on the guess $f = L_i$.

Definition 4.4 (the MDP $\mathcal{M}_i$, and Av_i) In Algorithm 3, the MDP $\mathcal{M}_i$ is given by $\mathcal{M}_{\mathrm{PlRd}}(\mathcal{G}, L_i) = \mathcal{M}(\mathcal{G}, \mathrm{Av}_{L_i})$. We write Av_i for available actions in

$\mathcal{M}_i$, that is, $\mathcal{M}_i = (S, \mathbf{1}, \mathbf{0}, A, \mathrm{Av}_i, \delta)$.

In the case of Algorithm 3, the MDPs $\mathcal{M}_0, \mathcal{M}_1, \dots$ do not only "converge" to $\mathcal{G}$, but also "reach $\mathcal{G}$ in finitely many steps," in the following sense.

Lemma 4.5 *In Algorithm 3, there exists $i_{\mathrm{M}} \in \mathbb{N}$ such that, for each $i \geq i_{\mathrm{M}}$, we have $V(\mathcal{G}) = V(\mathcal{M}_i)$.*

4.2 Local Propagation: from MDPs to WGs

Here is a technical observation that motivates the function $\mathcal{W}_{\mathrm{LcPg}}$.

Lemma 4.6 *Let $\mathcal{G}$ be the game in Algorithm 3, and $\mathrm{Av}' \colon S \to 2^A$ be a Minimizer restriction (Def. 4.2).*

1. *For each state $s \in S$, we have*
   $V(\mathcal{G})(s) \leq \max_{a \in \mathrm{Av}'(s)} \big(\mathbb{X}_a \big(V(\mathcal{G}) \big) \big)(s)$.
2. *For each $k \in \mathbb{N}$, we have*
   $$V(\mathcal{G})(s_0) \leq \max_{s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \cdots \xrightarrow{a_k} \text{ in } \mathrm{Av}'} \big(\mathbb{X}_{a_k} \big(V(\mathcal{G}) \big) \big)(s_k), \tag{6}$$
   where the maximum is taken over $a_0, s_1, a_1, \dots, s_k, a_k$ such that $a_0 \in \mathrm{Av}'(s_0), s_1 \in \mathrm{post}(s_0, a_0), a_1 \in \mathrm{Av}'(s_1), \dots, s_k \in \mathrm{post}(s_{k-1}, a_{k-1})$, $a_k \in \mathrm{Av}'(s_k)$.

Lem. 4.6.2, although not itself used in the following technical development, suggests the idea of global propagation for upper bounds. Note that a bound is given in (6) for each k; it is possible that a bound for some $k > 1$ is tighter than that for $k = 1$, motivating us to take a "look-ahead" further than one step.

However, the bound in (6) is not particularly tuned for tractability: computation of the maximum involves words whose number is exponential in k, and moreover, we want to do so for many k's.

In the end, our main technical contribution is that a similar "look-ahead" can be done by solving the widest path problem in the following weighted graph.

Definition 4.7 (the WG $\mathcal{W}_{\mathrm{LcPg}}(\mathcal{M}, f)$) Let $\mathcal{M} = (S, \mathbf{1}, \mathbf{0}, A, \mathrm{Av}', \delta)$ be an MDP, and $f \colon S \to [0, 1]$. The WG $\mathcal{W}_{\mathrm{LcPg}}(\mathcal{M}, f)$ is the following triple (S, E, w).

- Its set of vertices is S.
- We have $(s, s') \in E$ if and only if, for some $a \in \mathrm{Av}'(s)$, we have $s' \in \mathrm{post}(s, a)$ (i.e., $\delta(s, a, s') > 0$).
- The weight function $w \colon E \to [0, 1]$ is given by
  $$w(s, s') = \max \big\{ \mathbb{X}_a f(s) \;\big|\; a \in \mathrm{Av}'(s), s' \in \mathrm{post}(s, a) \big\}. \tag{7}$$

In (7), the function f—that is, the previous upper bound U_{i-1} in Algorithm 3—is propagated one step by the application of $\mathbb{X}_a$. This way of encoding these propagated values as weights in a WG seems pretty rough. For example, in case both s' and s'' are in $\mathrm{post}(s, a)$ for each $a \in \mathrm{Av}'(s)$, we have $w(s, s') = w(s, s'')$, no matter what the transition probabilities from s to s', s'' are. The return for this paid price (namely the information lost in the rough encoding) is that the resulting data structure (WG) allows fast *global* analysis via the widest path problem. Our experiment results in §5 demonstrate that this rough yet global approximation can make upper bounds quickly converge.

4.3 Soundness and Convergence

In Algorithm 3, an SG $\mathcal{G}$ is turned into an MDP $\mathcal{M}_i$ and then to a WG $\mathcal{W}_i$. Our claim is that computing a widest path in $\mathcal{W}_i$ gives the next upper bound U_i in the iteration. The following properties assure correctness and termination of Algorithm 3.

Theorem 4.8 (soundness) *In Algorithm 3, $V(\mathcal{G}) \leq$*

U_i *holds for each* $i \in \mathbb{N}$.

Theorem 4.9 (convergence) *In Algorithm 3, let the while loop iterate forever. Then* $U_i \to V(\mathcal{G})$ *as* $i \to \infty$.

5 Experiment Results

5.1 Experiment Settings

We compare the following four algorithms.

- *WP* is our BVI algorithm via widest paths. It avoids end component (EC) computation by global propagation of upper bounds.
- *DFL* is the implementation of the main algorithm in [20]. It relies on EC computation for deflating.
- *DFL_m* is our modification of DFL, where some unnecessary repetition of EC computation is removed.
- *DFL_BRTDP* is the learning-based variant of DFL. It restricts bound update to those states which are visited by simulations. See [20] for details.

The latter three—coming from [20]—are the only existing BVI algorithms for SGs with a convergence guarantee, to the best of our knowledge. The implementation of DFL and DFL_BRTDP is provided by the authors of [20].

The four algorithms are implemented on top of PRISM-games [21] version 2.0. We used the stopping threshold $\varepsilon = 10^{-6}$. The experiments were conducted on Dell Inspiron 3421 Laptop with 4.00 GB RAM and Intel(R) Core(TM) i5-3337U 1.80 GHz processor.

In the implementations of DFL and DFL_BRTDP, the deflating operation is applied only once every five iterations [20, §B.3]. Following this, our WP also solves the widest path problem (Line 8) only once every five iterations, while other operations are applied in each iteration.

For input SGs, we took four models from the literature: *mdsm* [11], *cloud* [6], *teamform* [12] and *investor* [22]. In addition, we used our model *manyECs*—an artificial model with many ECs—to assess the effect of ECs on performance. The model manyECs is presented in the appendix. Each of these five models comes with a model parameter N.

There is another model called *cdmsn* in [20]. We do not discuss cdmsn since all the algorithms (ours and those from [20]) terminated within 0.001 seconds.

5.2 Results

The number i of iterations and the running time for each algorithm and each input SG is shown in Table 1. For DFL_BRTDP, the ratio of states visited by the algorithm is shown in percentage; the smaller it is, the more efficient the algorithm is in reducing the state space. Each number for DFL_BRTDP (a probabilistic algorithm) is the average over 5 runs.

5.3 Discussion

We observe consistent performance advantage of our algorithm (WP). Even in the mdsm model where the DFL algorithms do not suffer from EC computation (#EC is just 1), WP's performance is comparable to DFL. The cloud model is where the learning-based approach in [20] works well—see visit% that are very small. Our WP performs comparably against DFL_BRTDP, too.

The performance advantage of our WP algorithm is eminent, not only in the artificial model of manyECs (where WP is faster by magnitudes), but also in the realistic model investor that comes from a financial application scenario [22]. The results for these two models suggest that WP is indeed advantageous when EC computation poses a bottleneck for other algorithms.

Overall, we observe that our WP algorithm can

表 1: Experimental results, comparing WP (our algorithm) with those in [20]. N is a model parameter (the bigger the more complex). #states, #trans, #EC show the numbers of states, transitions and ECs in the SG, respectively. itr is the number i of iterations at termination; time is the execution time in seconds. For each SG, the fastest algorithm is shaded in green. The settings that did not terminate are shaded in gray; TO is time out (6 hours), OOM is out of memory, and SO is stack overflow.

model	N	#states	#trans	#EC	DFL		DFL_m		DFL_BRTDP			WP	
					itr	time	itr	time	itr	visit%	time	itr	time
mdsm	3	62245	151143	1	121	3	121	4	17339	49.3	15	120	5
	4	335211	882765	1	125	15	125	47	91301	42.1	86	124	38
cloud	5	8842	60437	4421	7	7	7	1	167	6.9	14	7	<1
	6	34954	274965	17477	11	177	11	5	41	0.6	3	11	1
	7	139402	1237525	69701	11	19721	11	62	41	0.2	4	11	5
teamform	3	12475	15228	2754	2	<1	2	<1	972	49.0	137	2	<1
	4	96665	116464	19800	2	<1	2	<1	4154	34.6	9603	2	<1
	5	907993	1084752	176760	2	<1	2	<1			TO	2	<1
investor	50	211321	673810	29690	441	184	441	249			TO	364	48
	100	807521	2587510	114390	801	3318		OOM			TO	688	736
manyECs	500	1004	3007	502	6	7	6	7			TO	5	<1
	1000	2004	6007	1002	6	51	6	51			TO	5	<1
	5000	10004	30007	5002		SO		SO			TO	5	<1

be the first choice when it comes to solving SGs: for some models, it runs much faster than other algorithms; for other models, even if the performances of other algorithms differs a lot, WP's performance is comparable with the best algorithm.

6 Conclusions and Future Work

In this paper, we presented a new BVI algorithm for solving stochastic games. It features global propagation of upper bounds by widest paths, via a novel encoding of the problem to a suitable weighted graph. This way we avoid computation of end components that often penalizes the performance of the other BVI-based algorithms. Our experimental comparison with known BVI algorithms for SGs demonstrates the efficiency of our algorithm. For correctness of the algorithm, we presented proofs for soundness and convergence.

Extending the current algorithm for more advanced settings is future work—this is much like the results in [20] are extended and used in [2,3,16]. In doing so, we hope to make essential use of structures that are unique to those advanced problem settings. Another important direction is to push forward the idea of global propagation in verification and synthesis, seeking further instances of the idea. Finally, pursuing the global propagation idea in the context of reinforcement learning—where problems are often formalized using MDPs and the Bellman operator is heavily utilized—may open up another fruitful collaboration between formal methods and statistical machine learning.

Acknowledgment

The authors are supported by ERATO HASUO Metamathematics for Systems Design Project (No. JPMJER1603), JST; I.H. is supported by Grant-in-Aid No. 15KT0012, JSPS. Thanks are due to Maximilian Weininger and Edon Kelmendi for sharing their implementation, and to Pranav Ashok and

David Sprunger for useful discussions and comments.

参 考 文 献

[1] Andersson, D. and Miltersen, P. B.: The Complexity of Solving Stochastic Games on Graphs, *Algorithms and Computation, 20th International Symposium, ISAAC 2009, Honolulu, Hawaii, USA, December 16-18, 2009. Proceedings*, Dong, Y., Du, D., and Ibarra, O. H.(eds.), Lecture Notes in Computer Science, Vol. 5878, Springer, 2009, pp. 112–121.

[2] Ashok, P., Kretinsky, J., and Weininger, M.: Approximating Values of Generalized-Reachability Stochastic Games, *CoRR*, Vol. abs/1908.05106(2019).

[3] Ashok, P., Kretínský, J., and Weininger, M.: PAC Statistical Model Checking for Markov Decision Processes and Stochastic Games, *Computer Aided Verification - 31st International Conference, CAV 2019, New York City, NY, USA, July 15-18, 2019, Proceedings, Part I*, Dillig, I. and Tasiran, S.(eds.), Lecture Notes in Computer Science, Vol. 11561, Springer, 2019, pp. 497–519.

[4] Baier, C. and Katoen, J.-P.: *Principles of model checking*, MIT press, 2008.

[5] Brázdil, T., Chatterjee, K., Chmelik, M., Forejt, V., Kretínský, J., Kwiatkowska, M. Z., Parker, D., and Ujma, M.: Verification of Markov Decision Processes Using Learning Algorithms, *Automated Technology for Verification and Analysis - 12th International Symposium, ATVA 2014, Sydney, NSW, Australia, November 3-7, 2014, Proceedings*, Cassez, F. and Raskin, J.(eds.), Lecture Notes in Computer Science, Vol. 8837, Springer, 2014, pp. 98–114.

[6] Calinescu, R., Kikuchi, S., and Johnson, K.: Compositional Reverification of Probabilistic Safety Properties for Large-Scale Complex IT Systems, *Large-Scale Complex IT Systems. Development, Operation and Management - 17th Monterey Workshop 2012, Oxford, UK, March 19-21, 2012, Revised Selected Papers*, Calinescu, R. and Garlan, D.(eds.), Lecture Notes in Computer Science, Vol. 7539, Springer, 2012, pp. 303–329.

[7] Chatterjee, K., Dvorák, W., Henzinger, M., and Svozil, A.: Near-Linear Time Algorithms for Streett Objectives in Graphs and MDPs, *30th International Conference on Concurrency Theory, CONCUR 2019, August 27-30, 2019, Amsterdam, the Netherlands.*, Fokkink, W. and van Glabbeek, R.(eds.), LIPIcs, Vol. 140, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019, pp. 7:1–7:16.

[8] Chatterjee, K. and Fijalkow, N.: A reduction from parity games to simple stochastic games, *Proceedings of Second International Symposium on Games, Automata, Logics and Formal Verification, GandALF 2011, Minori, Italy, 15-17th June 2011*, D'Agostino, G. and La Torre, S.(eds.), EPTCS, Vol. 54, 2011, pp. 74–86.

[9] Chatterjee, K. and Henzinger, M.: Efficient and dynamic algorithms for alternating büchi games and maximal end-component decomposition, *Journal of the ACM (JACM)*, Vol. 61, No. 3(2014), pp. 15.

[10] Chatterjee, K. and Henzinger, T. A.: Value Iteration, *25 Years of Model Checking - History, Achievements, Perspectives*, Grumberg, O. and Veith, H.(eds.), Lecture Notes in Computer Science, Vol. 5000, Springer, 2008, pp. 107–138.

[11] Chen, T., Forejt, V., Kwiatkowska, M. Z., Parker, D., and Simaitis, A.: Automatic verification of competitive stochastic systems, *Formal Methods in System Design*, Vol. 43, No. 1(2013), pp. 61–92.

[12] Chen, T., Kwiatkowska, M. Z., Parker, D., and Simaitis, A.: Verifying Team Formation Protocols with Probabilistic Model Checking, *Computational Logic in Multi-Agent Systems - 12th International Workshop, CLIMA XII, Barcelona, Spain, July 17-18, 2011. Proceedings*, Leite, J., Torroni, P., Ågotnes, T., Boella, G., and van der Torre, L.(eds.), Lecture Notes in Computer Science, Vol. 6814, Springer, 2011, pp. 190–207.

[13] Condon, A.: The Complexity of Stochastic Games, *Inf. Comput.*, Vol. 96, No. 2(1992), pp. 203–224.

[14] Courcoubetis, C. and Yannakakis, M.: The Complexity of Probabilistic Verification, *J. ACM*, Vol. 42, No. 4(1995), pp. 857–907.

[15] De Alfaro, L.: *Formal verification of probabilistic systems*, Citeseer, 1997.

[16] Eisentraut, J., Kretinsky, J., and Rotar, A.: Stopping Criteria for Value and Strategy Iteration on Concurrent Stochastic Reachability Games, *CoRR*, Vol. abs/1909.08348(2019).

[17] Fredman, M. L. and Tarjan, R. E.: Fibonacci heaps and their uses in improved network optimization algorithms, *J. ACM*, Vol. 34, No. 3(1987), pp. 596–615.

[18] Haddad, S. and Monmege, B.: Interval iteration algorithm for MDPs and IMDPs, *Theoretical Computer Science*, Vol. 735(2018), pp. 111–131.

[19] Hoffman, A. J. and Karp, R. M.: On nonterminating stochastic games, *Management Science*, Vol. 12, No. 5(1966), pp. 359–370.

[20] Kelmendi, E., Krämer, J., Křetínskỳ, J., and Weininger, M.: Value iteration for simple stochastic games: stopping criterion and learning algorithm, *International Conference on Computer Aided Verification*, Springer, 2018, pp. 623–642.

[21] Kwiatkowska, M., Parker, D., and Wiltsche, C.: PRISM-games: Verification and Strategy Synthesis for Stochastic Multi-player Games with Multiple Objectives, *International Journal on Software Tools for Technology Transfer*, Vol. 20, No. 2(2017), pp. 195210.

[22] McIver, A. and Morgan, C.: Results on the

quantitative μ-calculus qMμ, *ACM Trans. Comput. Log.*, Vol. 8, No. 1(2007), pp. 3.

[23] McMahan, H. B., Likhachev, M., and Gordon, G. J.: Bounded real-time dynamic programming: RTDP with monotone upper bounds and performance guarantees, *Machine Learning, Proceedings of the Twenty-Second International Conference (ICML 2005), Bonn, Germany, August 7-11, 2005*, Raedt, L. D. and Wrobel, S.(eds.), ACM International Conference Proceeding Series, Vol. 119, ACM, 2005, pp. 569–576.

[24] Phalakarn, K., Takisaka, T., Haas, T., and Hasuo, I.: Widest Paths and Global Propagation in Bounded Value Iteration for Stochastic Games, *Computer Aided Verification - 32nd International Conference, CAV 2020, Los Angeles, CA, USA, July 21-24, 2020, Proceedings, Part II*, Lahiri, S. K. and Wang, C.(eds.), Lecture Notes in Computer Science, Vol. 12225, Springer, 2020, pp. 349–371.

[25] Svorenová, M. and Kwiatkowska, M.: Quantitative verification and strategy synthesis for stochastic games, *Eur. J. Control*, Vol. 30(2016), pp. 15–30.

[26] Ujma, M.: *On Verication and Controller Synthesis for Probabilistic Systems at Runtime*, PhD Thesis, Wolfson College, Univ. Oxford, 2015.