An Algorithm for Probabilistic Alternating Simulation

Chenyi Zhang[1,2] and Jun Pang[3]

[1] University of Queensland, Brisbane, Australia
[2] University of New South Wales, Sydney, Australia
[3] University of Luxembourg, Luxembourg

Abstract. In probabilistic game structures, probabilistic alternating simulation (PA-simulation) relations preserve formulas defined in probabilistic alternating-time temporal logic with respect to the behaviour of a subset of players. We propose a partition based algorithm for computing the largest PA-simulation. It is to our knowledge the first such algorithm that works in polynomial time. Our solution extends the generalised coarsest partition problem (GCPP) to a game-based setting with mixed strategies. The algorithm has higher complexities than those in the literature for non-probabilistic simulation and probabilistic simulation without mixed actions, but slightly improves the existing result for computing probabilistic simulation with respect to mixed actions.

1 Introduction

Simulation and bisimulation relations are useful tools in the verification of finite and infinite state systems. State space minimisation modulo these relations is a valuable technique to fight the state explosion problem in model checking, since bisimulation preserves properties formulated in logics like CTL and CTL* [8] while simulation preserves the universal (or safe) fragment of these logics [14].

In some situations, however, it is necessary to model quantitative aspects of a system. It is the case, for instance, in wireless networks, where we often need to assume that there is a chance of connection failure with a given rate. This requires modelling network systems with randomised behaviours (e.g., by pooling a connection after uncertain amount of time to minimise conflict). Another important fact of real-world systems is that environment changes, such as unexpected power-off, are often unpredictable. Therefore, we need to encode appropriate system behaviours to handle such situations, and in order to do so, it is sometimes crucial to employ probabilistic strategies to achieve the best possible outcomes [24]. One simple example is the rock-scissor-paper game where there is no deterministic strategy to win since the other player's move is unknown, but there is a probabilistic strategy, sometimes called *mixed strategy*, to win at least a third of all cases in a row, regardless of what the other player does.[4]

[4] A mixed strategy also ensures an eventual win but deterministic strategies do not.

A probabilistic game structure (PGS) is a model that has probabilistic transitions, and allows the consideration of probabilistic choices of players. The simulation relation in PGSs, called probabilistic alternating simulation (PA-simulation), has been shown to preserve a fragment of probabilistic alternating-time temporal logic (PATL) under *mixed strategies*, which is used in characterising what a group of players can enforce in such systems [25]. In this paper we propose a polynomial-time algorithm for computing the largest PA-simulation, which is, to the best of our knowledge, the first algorithm for computing a simulation relation in probabilistic concurrent games. A PGS combines the modelling of probabilistic transitions from probabilistic automata (PA), and the user interactions from concurrent game structures (GS). In PA, the probabilistic notions of simulation preserve PCTL safety formulas [20]. The *alternating simulation* [2] in GS has been been proved to preserve a fragment of ATL*, under the semantics of *deterministic strategies*. These simulation relations are computable in polynomial time for finite systems [26, 2].

Related work. Efficient algorithms have been proposed for computing the largest simulation (e.g., see [15, 22, 4, 13, 23]) in finite systems, with a variety of time and space complexities. In particular, Gentilini et al. [13] developed an efficient algorithm with an improved time complexity based on the work of Henzinger et al. [15] without losing the optimal space complexity. Van Glabbeek and Ploeger [23] later found a flaw in [13] and proposed a non-trivial fix. So far the best algorithm for time complexity is [18]. To compute probabilistic simulation, Baier et al. [3] reduce the problem of establishing a weight function for the lifted relation to a maximal flow problem. Cattani and Segala [5] reduce the problem of deciding strong probabilistic bisimulation to LP problems. Zhang and Hermanns [27] develop algorithms with improved time complexity for probabilistic simulations, following [3, 5]. A space efficient probabilistic simulation algorithm is proposed by Zhang [26] using the techniques proposed in [13, 23].

Studies on stochastic games have actually been carried out since as early as the 1950s [21], and a rich literature has developed in recent years (e.g. see [10, 9, 11, 6]). One existing approach called game metrics [12] defines approximation-based simulation relations, with a kernel simulation characterising the logic quantitative game μ-calculus [9], an extension of modal μ-calculus for concurrent games where each state is assigned a quantitative value in $[0, 1]$ for every formula. However, so far the best solutions in the literature on approximating the simulation as defined in the metrics for concurrent games potentially take exponential time [7]. Although PA-simulation is strictly stronger than the kernel simulation relation of the game metrics in [12], the algorithm presented in the paper has a more tractable complexity result, and we believe that it will benefit the abstraction or refinement based techniques for verifying game-based properties.

2 Preliminaries

Probabilistic game structures are defined in terms of discrete probabilistic distributions. A *discrete probabilistic distribution* Δ over a finite set S is a function of type $S \to [0,1]$, where $\sum_{s \in S} \Delta(s) = 1$. We write $\mathcal{D}(S)$ for the set of all such distributions on a fixed S. For a set $T \subseteq S$, define $\Delta(T) = \sum_{s \in T} \Delta(s)$. Given a finite index set I, a list of distributions $(\Delta_i)_{i \in I}$ and a list of probabilities $(p_i)_{i \in I}$ where, for all $i \in I$, $p_i \in [0,1]$ and $\sum_{i \in I} p_i = 1$, $\sum_{i \in I} p_i \Delta_i$ is obviously also a distribution. For $s \in S$, $\overline{s}$ is called a *point (or Dirac) distribution* satisfying $\overline{s}(s) = 1$ and $\overline{s}(t) = 0$ for all $t \neq s$. Given $\Delta \in \mathcal{D}(S)$, we define $\lceil \Delta \rceil$ as the set $\{s \in S \mid \Delta(s) > 0\}$, which is the *support* of Δ.

In this paper we assume a set of two players $\{\mathtt{I}, \mathtt{II}\}$ (though our results can be extended to handle a finite set of players as in the standard game structure and ATL semantics [1]), and *Prop* a finite set of propositions.

Definition 1. *A probabilistic game structure $\mathcal{G}$ is a tuple $\langle S, s_0, \mathcal{L}, Act, \delta \rangle$, where*

- *S is a finite set of states, with s_0 the initial state;*
- *$\mathcal{L} : S \to 2^{Prop}$ is the labelling function which assigns to each state $s \in S$ a set of propositions that are true in s;*
- *$Act = Act_{\mathtt{I}} \times Act_{\mathtt{II}}$ is a finite set of joint actions, where $Act_{\mathtt{I}}$ and $Act_{\mathtt{II}}$ are, respectively, the sets of actions for players $\mathtt{I}$ and $\mathtt{II}$;*
- *$\delta : S \times Act \to \mathcal{D}(S)$ is a transition function.*

If in state s player $\mathtt{I}$ performs action a_1 and player $\mathtt{II}$ performs action a_2 then $\delta(s, \langle a_1, a_2 \rangle)$ is the distribution for the next states. During each step the players choose their next moves simultaneously. We define a *mixed action* of player $\mathtt{I}$ ($\mathtt{II}$) as a distribution over $Act_{\mathtt{I}}$ ($Act_{\mathtt{II}}$), and write $\Pi_{\mathtt{I}}$ ($\Pi_{\mathtt{II}}$) for the set of mixed actions of player $\mathtt{I}$ ($\mathtt{II}$).[5] In particular, $\overline{a}$ is a *deterministic* mixed action which always chooses a. We lift the transition function δ to handle mixed actions. Given $\pi_1 \in \Pi_{\mathtt{I}}$ and $\pi_2 \in \Pi_{\mathtt{II}}$, for all $s, t \in S$, we have

$$\overline{\delta}(s, \langle \pi_1, \pi_2 \rangle)(t) = \sum_{a_1 \in Act_{\mathtt{I}}, a_2 \in Act_{\mathtt{II}}} \pi_1(a_1) \cdot \pi_2(a_2) \cdot \delta(s, \langle a_1, a_2 \rangle)(t)$$

Simulation relations in probabilistic systems require a definition of *lifting* [16], which extends the relations to the domain of distributions.[6] Let S, T be two sets and $\mathcal{R} \subseteq S \times T$ be a relation, then $\overline{\mathcal{R}} \subseteq \mathcal{D}(S) \times \mathcal{D}(T)$ is a *lifted relation* defined by $\Delta \, \overline{\mathcal{R}} \, \Theta$ if there exists a weight function $w : S \times T \to [0,1]$ such that (1) $\sum_{t \in T} w(s,t) = \Delta(s)$ for all $s \in S$, (2) $\sum_{s \in S} w(s,t) = \Theta(t)$ for all $t \in T$, (3) $s \mathcal{R} t$ for all $s \in S$ and $t \in T$ with $w(s,t) > 0$.

The intuition behind the lifting is that each state in the support of one distribution may correspond to a number of states in the support of the other distribution, and vice versa. The example in Fig. 1 is taken from [19] to show how to lift one relation. We have two set of states $S = \{s_1, s_2\}$ and $T = \{t_1, t_2, t_3\}$, and $\mathcal{R} = \{(s_1, t_1), (s_1, t_2), (s_2, t_2), (s_2, t_3)\}$. We have $\Delta \, \overline{\mathcal{R}} \, \Theta$, where $\Delta(s_1) = \Delta(s_2) =$

[5] Note $\Pi_{\mathtt{I}}$ is equivalent to $\mathcal{D}(Act_{\mathtt{I}})$, though we choose a different symbol because the origin of a mixed action is a simplified *mixed strategy* of player $\mathtt{I}$ which has type $S^+ \to \mathcal{D}(Act_{\mathtt{I}})$. A mixed action only considers player $\mathtt{I}$'s current step.

[6] In a probabilistic system without explicit user interactions, state s is simulated by state t if for every $s \xrightarrow{a} \Delta_1$ there exists $t \xrightarrow{a} \Delta_2$ such that Δ_1 is simulated by Δ_2.

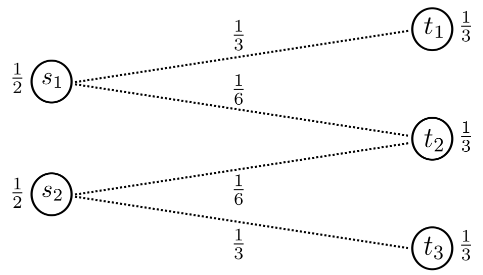

Fig. 1. An example showing how to lift one relation.

$\frac{1}{2}$ and $\Theta(t_1) = \Theta(t_2) = \Theta(t_3) = \frac{1}{3}$. To check this, we define a weight function w by: $w(s_1, t_1) = \frac{1}{3}$, $w(s_1, t_2) = \frac{1}{6}$ $w(s_2, t_2) = \frac{1}{6}$, and $w(s_2, t_3) = \frac{1}{3}$. The dotted lines indicate the allocation of weights required to relate Δ to Θ via $\overline{\mathcal{R}}$. By lifting in this way, we are able to extend the notion of alternating simulation [2] to a probabilistic setting.

Definition 2. *Given a PGS $\langle S, s_0, \mathcal{L}, Act, \delta\rangle$, a* probabilistic alternating I-simulation *(PA-I-simulation) is a relation $\sqsubseteq\ \subseteq S \times S$ such that if $s \sqsubseteq t$, then*

- *$\mathcal{L}(s) = \mathcal{L}(t)$,*
- *for all $\pi_1 \in \Pi_{\mathtt{I}}$, there exists $\pi'_1 \in \Pi_{\mathtt{I}}$, such that for all $\pi'_2 \in \Pi_{\mathtt{II}}$, there exists $\pi_2 \in \Pi_{\mathtt{II}}$, such that $\overline{\delta}(s, \langle\pi_1, \pi_2\rangle) \mathrel{\overline{\sqsubseteq}} \overline{\delta}(t, \langle\pi'_1, \pi'_2\rangle)$.*

If s PA-I-simulates t and t PA-I-simulates s, we say s and t are *PA-I-simulation equivalent.*[7]

PA-I-simulation has been shown to preserve a fragment of PATL which covers the ability of player I to enforce certain temporal requirements [25]. For example, if in state s player I can enforce reaching some states satisfying p within 5 transition steps and with probability at least $\frac{1}{2}$, written $s \models \langle\langle \mathtt{I}\rangle\rangle^{\geq\frac{1}{2}} \Diamond^{\leq 5} p$, then for every state t that simulates s with respect to I, i.e., $s \sqsubseteq t$ by some PA-I-simulation '$\sqsubseteq$', we also have $t \models \langle\langle \mathtt{I}\rangle\rangle^{\geq\frac{1}{2}} \Diamond^{\leq 5} p$.

General Coarsest Partition Problem

The general coarsest partition problem (GCPP) provides a characterisation of (non-probabilistic) simulation in finite state transition systems [13]. Informally, in this approach, states that are (non-probabilistic) simulation equivalent are grouped into the same block, and all such blocks form a partition over the (finite) state space. Based on the partition, blocks are further related by a partial order $\preceq$, so that if $P \preceq Q$, then every state in block P is simulated by every state in block Q. The GCPP is to find, for a given PGS, the smallest such set of blocks. In the literature such a methodology yields space efficient algorithms for computing the largest (non-probabilistic) simulation relation in a finite system [13, 23]. Similar methods have been adopted and developed to compute the largest simulation relations in the model of probabilistic automata [26].

[7] Alternating simulations and equivalences are for player I unless stated otherwise.

We briefly review the basic notions that are required to present the GCPP problem. A *partition* over a set S, is a collection $\Sigma \subseteq \mathcal{P}(S)$ satisfying (1) $\bigcup \Sigma = S$ and (2) $P \cap Q = \emptyset$ for all distinct *blocks* $P, Q \in \Sigma$. Given $s \in S$, write $[s]_\Sigma$ for the block in partition Σ that contains s. A partition Σ_1 is *finer* than Σ_2, written $\Sigma_1 \lhd \Sigma_2$, if for all $P \in \Sigma_1$ there exists $Q \in \Sigma_2$ such that $P \subseteq Q$.

Given a set S, a *partition pair* over S is $(\Sigma, \preceq)$ where Σ is a partition over S and $\preceq \subseteq \Sigma \times \Sigma$ is a partial order. Write $Part(S)$ for the set of partition pairs on S. If $\Upsilon \lhd \Sigma$ and $\preceq$ is a relation on Σ, then $\preceq(\Upsilon) = \{(P, Q) \mid P, Q \in \Upsilon, \exists P', Q' \in \Sigma, P \subseteq P', Q \subseteq Q', P' \preceq Q'\}$ is the relation on Υ *induced* by $\preceq$. Let $(\Sigma_1, \preceq_1)$ and $(\Sigma_2, \preceq_2)$ be partition orders, write $(\Sigma_1, \preceq_1) \leq (\Sigma_2, \preceq_2)$ if $\Sigma_1 \lhd \Sigma_2$, and $\preceq_1 \subseteq \preceq_2 (\Sigma_1)$. Define a relation $\sqsubseteq_{(\Sigma,\preceq)} \subseteq S \times S$ as determined by a partition pair $(\Sigma, \preceq)$ by $s \sqsubseteq_{(\Sigma,\preceq)} t$ iff $[s]_\Sigma \preceq [t]_\Sigma$.

Let $\rightarrow \subseteq S \times S$ be a (transition) relation and $\mathcal{L} : S \rightarrow 2^{Prop}$ a labelling function, then a relation $\sqsubseteq$ is a simulation on S if for all $s, t \in S$ with $s \sqsubseteq t$, we have (1) $\mathcal{L}(s) = \mathcal{L}(t)$ and (2) $s \rightarrow s'$ implies that there exists t' such that $t \rightarrow t'$ and $s' \sqsubseteq t'$. Let $(\Sigma, \preceq)$ be a partition pair on S, then it is *stable* with respect to $\rightarrow$ if for all $P, Q \in \Sigma$ with $P \preceq Q$ and $s \in P$ such that $s \rightarrow s'$ with $s' \in P' \in \Sigma$, then there exists $Q' \in \Sigma$ such that $P' \preceq Q'$ and for all $t \in Q$, there exists $t' \in Q'$ such that $t \rightarrow t'$. The following result is essential to the GCPP approach, as we derive the largest simulation relation by computing the coarsest stable partition pair over a finite state space.[8]

Proposition 1. *[13, 23] Let $(\Sigma, \preceq)$ be a partition pair, then it is stable with respect to $\rightarrow$ iff the induced relation $\sqsubseteq_{(\Sigma,\preceq)}$ is a simulation (with respect to $\rightarrow$).*

Given a transition relation on a state space there exists a unique largest simulation relation. Thus, solutions to GCPP provide the coarsest stable partition pairs, and they have been proved to characterise the largest simulation relations in non-probabilistic systems [13, 23].

3 Solving GCPP in Probabilistic Game Structures

In this section we extend the GCPP framework to characterise PA-simulations in PGSs. Given a PGS $\mathcal{G} = \langle S, s_0, \mathcal{L}, Act, \delta \rangle$, a *partition pair* over $\mathcal{G}$ is $(\Sigma, \preceq)$ where Σ is a partition over S. Write $Part(\mathcal{G})$ for the set of all partition pairs over S. We show how to compute the coarsest partition pair and prove that it characterises the largest PA-simulation for a given player.

Since in probabilistic systems transitions go from states to distributions over states, we first present a probabilistic version of *stability*, as per [26]. Let $\rightarrow \subseteq S \times \mathcal{D}(S)$ be a probabilistic (transition) relation. For a distribution $\Delta \in \mathcal{D}(S)$ and Σ a partition, write Δ_Σ as a distribution on Σ defined by $\Delta_\Sigma(P) = \Delta(P)$ for all $P \in \Sigma$. Let $(\Sigma, \preceq)$ be a partition pair, it is *stable* with respect to the

[8] We choose the word *coarsest* for partition pairs to make it consistent with the standard term GCPP, and it is clear in the context that *coarsest* carries the same meaning as *largest* with respect to the order $\leq$ defined on partition pairs.

relation $\to$, if for all $P, Q \in \Sigma$ with $P \preceq Q$ and $s \in P$ such that $s \to \Delta$, then for all $t \in Q$ there exists $t \to \Theta$ such that $\Delta_\Sigma \overline{\preceq} \Theta_\Sigma$.

Another obstacle in characterising PA-simulation is that the concerned player can only partially determine a transition. That is, after player `I` performs an action on a state, the exact future distribution on next states depends on an action from player `II`. Therefore, we need to (again) lift the stability condition for PA-`I`-simulation from distributions to sets of distributions.

Let $\leq \subseteq S \times S$ be a partial order on a set S, define $\leq_{Sm} \subseteq \mathcal{P}(S) \times \mathcal{P}(S)$, by $P \leq_{Sm} Q$ if for all $t \in Q$ there exists $s \in P$ such that $s \leq t$. In the literature this definition is known as a 'Smyth order'. In a PGS, we 'curry' the transition function by defining $\overline{\delta}(s, \pi_1) = \{\overline{\delta}(s, \langle \pi_1, \pi_2 \rangle) \mid \pi_2 \in \Pi_{\mathtt{II}}\}$, which is the set of distributions that are possible if player `I` takes a mixed action $\pi_1 \in \Pi_{\mathtt{I}}$ on $s \in S$.

Definition 3. *(lifted stability) Let $(\Sigma, \preceq)$ be a partition pair on S in a PGS, it is stable with respect to player `I`'s choice, if for all $\pi \in \Pi_I$, $P, Q \in \Sigma$ with $P \preceq Q$ and $s \in P$, there exists $\pi' \in \Pi_I$ such that $\overline{\delta}(s, \pi)_\Sigma \overline{\preceq}_{Sm} \overline{\delta}(t, \pi')_\Sigma$ for all $t \in Q$.*

Intuitively, the Smyth order captures the way of *behavioral* simulation. That is, if $\overline{\delta}(t, \pi')$ is at least as restrictive as $\overline{\delta}(s, \pi)$, then whatever player `I` is able to enforce by performing π in s, he can also enforce it by performing π' in t, as player `II` has *fewer* choices in $\overline{\delta}(t, \pi')$ than in $\overline{\delta}(s, \pi)$. At this point, for the sake of readability, if it is clear from the context, we write W for W_Σ as the distribution W mapped onto partition Σ.

For simulation relations, it is also required that the related states agree on their labelling. Define Σ_0 as the *labelling partition* satisfying for all $s, t \in S$, $\mathcal{L}(s) = \mathcal{L}(t)$ iff $[s]_{\Sigma_0} = [t]_{\Sigma_0}$. Write $Part^0(\mathcal{G}) \subseteq Part(\mathcal{G})$ for the set of partition pairs $(\Sigma, \preceq)$ satisfying $(\Sigma, \preceq) \leq (\Sigma_0, \mathtt{Id})$, where `Id` is the identity relation.

Lemma 1. *For all $(\Sigma, \preceq) \in Part^0(\mathcal{G})$, if $(\Sigma, \preceq)$ is a stable partition pair with respect to player `I`'s choice then $\sqsubseteq_{(\Sigma, \preceq)}$ is a PA-`I`-simulation.*

Obviously every PA-`I`-simulation is contained in the relation induced by $(\Sigma_0, \mathtt{Id})$, and moreover, the above lemma asserts that every stable partition pair smaller than $(\Sigma_0, \mathtt{Id})$ is a PA-`I`-simulation. In the following, we try to compute the coarsest partition pair by refining $(\Sigma_0, \mathtt{Id})$ until it stabilises. The resulting stable partition pair can be proved to characterise the largest PA-`I`-simulation on the state space S as required.

We say t simulates s with respect to player-`I`'s choice on a partition pair $(\Sigma, \preceq)$ if for all $\pi \in \Pi_{\mathtt{I}}$, there exists $\pi' \in \Pi_{\mathtt{I}}$ such that $\overline{\delta}(s, \pi) \overline{\preceq}_{Sm} \overline{\delta}(t, \pi')$. For better readability, sometimes we also say t simulates s on $(\Sigma, \preceq)$ if it is clear from the context. Let $(\Sigma_1, \preceq_1) \leq (\Sigma_2, \preceq_2)$, we say $(\Sigma_1, \preceq_1)$ is stable on $(\Sigma_2, \preceq_2)$, if for all $P, Q \in \Sigma_1$ with $P \preceq_1 Q$, $s \in P$ and $t \in Q$, t simulates s on $(\Sigma_2, \preceq_2)$.

Definition 4. *Define an operator $\rho : Part(\mathcal{G}) \to Part(\mathcal{G})$, such that $\rho((\Sigma, \preceq))$ is the largest partition pair $(\Sigma', \preceq') \leq (\Sigma, \preceq)$ that is stable on $(\Sigma, \preceq)$.*

The operator ρ has the following properties.

Lemma 2. *1) ρ is well defined on $Part(\mathcal{G})$. 2) ρ is monotonic on $(Part^0(\mathcal{G}), \leq)$.*

Lemma 1 ensures that for all $(\Sigma, \preceq) \in Part^0(\mathcal{G})$, $\sqsubseteq_{(\Sigma,\preceq)}$ is a PA-$\mathtt{I}$-simulation if $\rho((\Sigma, \preceq)) = (\Sigma, \preceq)$, i.e., $(\Sigma, \preceq)$ is a fixpoint of ρ. However, we still need to find the largest PA-$\mathtt{I}$-simulation. The following result indicates that if S is finite, the coarsest stable partition pair achieved by repetitively applying ρ on $(\Sigma_0, \mathtt{Id})$ indeed yields the largest PA-$\mathtt{I}$-simulation.[9] Define $\rho^0(X) = X$ and $\rho^{n+1}(X) = \rho(\rho^n(X))$ for partition pairs X.

Theorem 1. *Let* $(\Sigma, \preceq) = \bigcap_{i\in\mathbb{N}} \rho^i((\Sigma_0, \mathtt{Id}))$*, then* $\sqsubseteq_{(\Sigma,\preceq)}$ *is the largest PA-*$\mathtt{I}$*-simulation on* $\mathcal{G}$*.*

Proof. (sketch) Let $\sqsubseteq^+$ be the largest PA-$\mathtt{I}$-simulation on $\mathcal{G}$. Define a set $\Sigma^+ = \{\{t \in S \mid s \sqsubseteq^+ t \wedge t \sqsubseteq^+ s\} \mid s \in S\}$. Since $\sqsubseteq^+$ is the largest PA-$\mathtt{I}$-simulation, it can be shown that $\sqsubseteq^+$ is reflexive, symmetric and transitive within each block $P \in \Sigma^+$. Moreover, we define a relation $\preceq^+$ by $P \preceq^+ Q$ if there exists $s \in P$ and $t \in Q$ such that $s \sqsubseteq^+ t$, and it can be shown that $\preceq^+$ is a partial order on Σ^+. Then $(\Sigma^+, \preceq^+)$ forms a partition pair on $\mathcal{G}$, and furthermore, it is stable, and we also have $(\Sigma^+, \preceq^+) \leq (\Sigma_0, \mathtt{Id})$.

We apply ρ on both sides. By Lemma 2(2) (monotonicity), and $(\Sigma^+, \preceq^+)$ being stable, we have $(\Sigma^+, \preceq^+) = \rho^i((\Sigma^+, \preceq^+)) \leq \rho^i((\Sigma_0, \mathtt{Id}))$ for all $i \in \mathbb{N}$. As $Part(\mathcal{G})$ is finite, there exists $j \in \mathbb{N}$, such that $\rho^j((\Sigma_0, \mathtt{Id})) = \rho^{j+1}((\Sigma_0, \mathtt{Id}))$. Therefore, $\rho^j((\Sigma_0, \mathtt{Id}))$ is a stable partition pair, and $\sqsubseteq_{\rho^j((\Sigma_0,\mathtt{Id}))}$ is a PA-$\mathtt{I}$-simulation by Lemma 1. Straightforwardly we have $\sqsubseteq^+ \subseteq \sqsubseteq_{\rho^j((\Sigma_0,\mathtt{Id}))}$. Since $\sqsubseteq^+$ is the largest PA-$\mathtt{I}$-simulation by assumption, we have $\sqsubseteq^+ = \sqsubseteq_{\rho^j((\Sigma_0,\mathtt{Id}))}$, and the result directly follows. $\square$

4 A Decision Procedure for PA-I-Simulation

Efficient algorithms for simulation in the non-probabilistic setting sometimes apply predecessor based methods [15, 13] for splitting blocks and refining partitions. This method can no longer be applied for simulations in the probabilistic setting, as the transition functions now map a state to a state distribution rather than a single state, and simulation relation needs to be *lifted* to handle distributions. The algorithms in [27, 26] follow the approaches in [3] by reducing the problem of deciding a weight function on lifted relations to checking the value of a maximal flow problem. This method, however, does *not* apply to combined transitions, where a more general solution is required. Algorithms for deciding probabilistic bisimulations [5] reduce the problem on checking weight functions with combined choices to solutions in linear programming (LP), which are known to be decidable in polynomial time [17].[10]

Simulation relations are characterised by partition pairs in the solutions to the GCPP. We propose the following characterisation of lifting in order to handle

[9] The following proof resembles the classical paradigm of finding the least fixpoint in an ω-chain of a complete partial order by treating $(\Sigma_0, \mathtt{Id})$ as $\perp$. However, here we also need that fixpoint to represent the largest PA-$\mathtt{I}$-simulation.

[10] The maximal flow problem is a special instance of an LP problem, which can be solved more efficiently.

the partial order relation on partitions. Let S be a finite set and $\preceq$ a partial order on S. Define $\lfloor s \rfloor_{\preceq} = \{t \in S \mid s \preceq t\}$, which is called the *up-closure* of s. The following lemma reduces the problems of finding a weight function for two distributions on a partition pair to comparing weights of each up-closed block, and the latter problem can be easily encoded in LP when checking PA-I-simulation on a given partition pair between two states (as shown in Lemma 7).

Lemma 3. *Let S be a set with a partial order $\preceq \subseteq S \times S$ and $\Delta_1, \Delta_2 \in \mathcal{D}(S)$, then $\Delta_1 \mathrel{\overline{\preceq}} \Delta_2$ iff we have $\Delta_1(\lfloor s \rfloor_{\preceq}) \leq \Delta_2(\lfloor s \rfloor_{\preceq})$ for all $s \in S$.*

When deciding whether s is able to simulate t with respect to I's choice on a certain partition pair, we need to examine potentially infinitely many mixed actions in $\Pi_{\mathtt{I}}$. This problem can be moderated by the following observations. First we show that for s to be simulated by t, it is only required to check all deterministic choices of player I on s.

Lemma 4. *Let $(\Sigma, \preceq)$ be a partition pair, then t simulates s on $(\Sigma, \preceq)$ if for all $a \in Act_I$, there exists $\pi \in \Pi_I$ such that $\overline{\delta}(s, \overline{a}) \mathrel{\overline{\preceq}_{Sm}} \overline{\delta}(t, \pi)$.*

The next lemma states that for checking a Smyth order $\overline{\delta}(s, \pi) \mathrel{\overline{\preceq}_{Sm}} \overline{\delta}(t, \pi')$, it suffices to focus on player II's deterministic choices in $\overline{\delta}(t, \pi')$, since all probabilistic choices can be represented as interpolations from deterministic choices.

Lemma 5. *$\overline{\delta}(s, \pi) \mathrel{\overline{\preceq}_{Sm}} \overline{\delta}(t, \pi')$ if for all $a \in Act_{II}$, there exists $\pi'' \in \Pi_{II}$ such that $\overline{\delta}(s, \langle \pi, \pi'' \rangle) \mathrel{\overline{\preceq}} \overline{\delta}(t, \langle \pi', \overline{a} \rangle)$.*

Combining the above two lemmas, we have the following.

Lemma 6. *Let $(\Sigma, \preceq)$ be a partition pair, then t simulates s with respect to player-I's choice on $(\Sigma, \preceq)$ if for all $a_1 \in Act_I$, there exists $\pi_1 \in \Pi_I$ such that for all $a_2 \in Act_{II}$, there exists $\pi_2 \in \Pi_{II}$ such that $\overline{\delta}(s, \langle \overline{a_1}, \pi_2 \rangle) \mathrel{\overline{\preceq}} \overline{\delta}(t, \langle \pi_1, \overline{a_2} \rangle)$.*

The following lemma states how to check if the action a can be followed by a mixed action from $\Pi_{\mathtt{I}}$.

Lemma 7. *Given a partition pair $(\Sigma, \preceq)$, two states $s, t \in S$ and $a \in Act_I$, there exists $\pi \in \Pi_I$ such that $\overline{\delta}(s, \overline{a}) \mathrel{\overline{\preceq}_{Sm}} \overline{\delta}(t, \pi)$, iff the following LP has a solution: Let $Act_I = \{a_1, a_2, \ldots, a_\ell\}$ and $Act_{II} = \{b_1, b_2, \ldots, b_m\}$*

$$\sum_{i=1}^{\ell} \alpha_i = 1 \tag{1}$$

$$\forall i = 1, 2, \ldots, \ell : 0 \leq \alpha_i \leq 1 \tag{2}$$

$$\forall j = 1, 2, \ldots, m : \sum_{k=1}^{m} \beta_{j,k} = 1 \tag{3}$$

$$\forall j, k = 1, 2, \ldots, m : 0 \leq \beta_{j,k} \leq 1 \tag{4}$$

$\forall B \in \Sigma : j = 1, 2, \ldots, m :$

$$\sum_{k=1}^{m} \beta_{j,k} \cdot \delta(s, a, b_k)(\lfloor B \rfloor_{\preceq}) \leq \sum_{i=1}^{\ell} \alpha_i \cdot \delta(t, a_i, b_j)(\lfloor B \rfloor_{\preceq}) \tag{5}$$

Here $\alpha_1, \alpha_2, \ldots, \alpha_\ell$ are used to 'guess' a mixed action from player $\mathtt{I}$, as constrained in Eq. 1 and Eq. 2. To establish the Smyth order $\overline{\preceq}_{Sm}$, by Lemma 6, for every player $\mathtt{II}$ action b_j with $j = 1, 2, \ldots, m$, we 'guess' a mixed action from $Act_{\mathtt{II}}$ represented by $\beta_{j,1}, \beta_{j,2} \ldots, \beta_{j,m}$, as constrained in Eq. 3 and Eq. 4. Then for each block B in Σ, the established distributions need to satisfy the lifted relation $\overline{\preceq}$, which is characterised by the inequalities on the up-closure of B with respect to the order $\preceq$, by Lemma 3.

We define a predicate $\mathsf{CanFollow}$ such that $\mathsf{CanFollow}((\Sigma, \preceq), s, t, a)$ decides whether there exists a mixed action of player $\mathtt{I}$ from t which simulates action $a \in Act_{\mathtt{I}}$ from s on the partition pair $(\Sigma, \preceq)$. $\mathsf{CanFollow}$ establishes an LP problem from its parameters (see Lemma 7). We further define a predicate CanSim which decides whether a state simulates another with respect to player $\mathtt{I}$'s choice on $(\Sigma, \preceq)$ for all actions in $Act_{\mathtt{I}}$, i.e., $\mathsf{CanSim}((\Sigma, \preceq), s, t)$ returns *true* if $\mathsf{CanFollow}((\Sigma, \preceq), s, t, a)$ returns *true* for all $a \in Act_{\mathtt{I}}$.

Algorithm 1 Refining a block to make it stable on a partition pair

```
INPUT: a partition pair $(\Sigma, \preceq)$, a block $B \in \Sigma$
OUTPUT: a partition pair $(\Sigma_B, \preceq_B)$ on $B$
function Split $((\Sigma, \preceq), B)$
   $\Sigma_B := \{\{s\} \mid s \in B\}$; $\preceq_B := \{(s, s) \mid s \in B\}$; $\Sigma' := \emptyset$; $\preceq' := \emptyset$
   while $\Sigma_B \neq \Sigma' \vee \preceq_B \neq \preceq'$ do
      $\Sigma' := \Sigma_B$; $\preceq' := \preceq_B$
      for each distinct $B_1, B_2 \in \Sigma_B$ do
         pick any $s_1 \in B_1$ and $s_2 \in B_2$
         if $(\mathsf{CanSim}((\Sigma, \preceq), s_1, s_2) \wedge \mathsf{CanSim}((\Sigma, \preceq), s_2, s_1))$ then
            $\Sigma_B := \Sigma_B \setminus \{B_1, B_2\} \cup \{B_1 \cup B_2\}$
            $\preceq_B := \preceq_B \cup \{(X, B_1 \cup B_2) \mid X \in \Sigma : (X, B_1) \in \preceq_B \vee (X, B_2) \in \preceq_B\}$
                  $\cup \{(B_1 \cup B_2, X) \mid X \in \Sigma : (B_1, X) \in \preceq_B \vee (B_2, X) \in \preceq_B\}$
                  $\setminus \{(B_i, X), (X, B_i) \mid X \in \Sigma : (B_i, X), (X, B_i) \in \preceq_B \wedge i \in \{1, 2\}\}$
         else if $(\mathsf{CanSim}((\Sigma, \preceq), s_1, s_2))$ then
            $\preceq_B := \preceq_B \cup \{(B_2, B_1)\}$
         else if $(\mathsf{CanSim}((\Sigma, \preceq), s_2, s_1))$ then
            $\preceq_B := \preceq_B \cup \{(B_1, B_2)\}$
      endfor
   endwhile
   return $(\Sigma_B, \preceq_B)$
```

Algorithm 1 defines a function Split which refines a block $B \in \Sigma$ into a partition pair corresponding the maximal simulation that is stable on $(\Sigma, \preceq)$. It starts with the finest partition and the identity relation (as the final relation is reflexive). For each pair of blocks in the partition, we check if they can simulate each other by picking up a state from each block. If they are simulation equivalent on $(\Sigma, \preceq)$ then we merge the two blocks as well as all incoming and outgoing relation in the current partial order. If only one simulates the other we add an appropriate pair into the current ordering. This process continues until the partition pair stablises.

Algorithm 2 Computing the Generalised Coarsest Partition Pair

```
INPUT: a probabilistic game structure $\mathcal{G} = \langle S, s_0, \mathcal{L}, Act, \delta \rangle$
OUTPUT: a partition pair $(\Sigma, \preceq)$ on $S$
function GCPP $(\mathcal{G})$
    $\Sigma := \{\{t \mid \mathcal{L}(t) = \mathcal{L}(s)\} \mid s \in S\}$; $\preceq := \{(B, B) \mid B \in \Sigma\}$
    $\Sigma' := \emptyset$; $\preceq' := \emptyset$
    while $\Sigma \neq \Sigma' \vee \preceq \neq \preceq'$ do
        $\Sigma' := \Sigma$; $\preceq' := \preceq$
        for each $B \in \Sigma$ do
            $(\Sigma_B, \preceq_B) := \mathsf{Split}((\Sigma', \preceq'), B)$
            $\Sigma := \Sigma \setminus \{B\} \cup \Sigma_B$
            $\preceq := \preceq \cup \preceq_B$
                $\cup \{(B', X) \mid X \in \Sigma : B' \in \Sigma_B : (B, X) \in \preceq\}$
                $\cup \{(X, B') \mid X \in \Sigma : B' \in \Sigma_B : (X, B) \in \preceq\}$
                $\setminus \{(B, X), (X, B) \mid X \in \Sigma : (X, B), (B, X) \in \preceq\}$
        endfor
    endwhile
    return $(\Sigma, \preceq)$
```

Algorithm 2 is based on the functionality of Split in Algorithm 1. Starting from the partition $(\Sigma_0, \mathtt{Id})$, which is identified as $(\{\{t \mid \mathcal{L}(t) = \mathcal{L}(s)\} \mid s \in S\}, \{(B, B) \mid B \in \Sigma_0\})$, the algorithm computes a sequence of partition pairs $(\Sigma_1, \preceq_1), (\Sigma_2, \preceq_2) \ldots$ until it stabilises, which is detected by checking the condition $\Sigma \neq \Sigma' \vee \preceq \neq \preceq'$. At each iteration we have $(\Sigma_{i+1}, \preceq_{i+1}) \leq (\Sigma_i, \preceq_i)$, and moreover, $(\Sigma_{i+1}, \preceq_{i+1})$ is the maximal partition pair that is stable on $(\Sigma_i, \preceq_i)$. The correctness of the algorithm is justified by Theorem 1, which states that it converges to the coarsest partition pair that is contained in $(\Sigma_0, \mathtt{Id})$ and returns a representation of the largest PA-$\mathtt{I}$-simulation.

Space complexity. For a PGS $\langle S, s_0, \mathcal{L}, Act, \delta \rangle$, it requires $\mathcal{O}(|S|)$ to store the state space and $\mathcal{O}(|S|^2 \cdot |Act|)$ for the transition relation, since for each $s \in S$ and $\langle a_1, a_2 \rangle \in Act$ it requires an array of size $\mathcal{O}(|S|)$ to store a distribution. Recording a partition pair takes $\mathcal{O}(|S| \log |S| + |S|^2)$ as the first part is needed to record for each state which equivalence class in the partition it belongs, and the second part is needed for the partial order relation $\preceq$ which takes at most $\mathcal{O}(|S|^2)$. The computation from $(\Sigma_i, \preceq_i)$ to $(\Sigma_{i+1}, \preceq_{i+1})$ can be done in-place which only requires additional constant space to track if the partition pair has been modified during each iteration. Another extra space-consuming part is for solving LP constrains, which we assume has space usage $\mathcal{O}(\gamma(N))$ where $N = 1 + |Act_{\mathtt{I}}| + |Act_{\mathtt{II}}| + |Act_{\mathtt{II}}|^2 + |S| \cdot |Act_{\mathtt{II}}|$ is the number of linear constraints at most, and $\gamma(N)$ some polynomial. The space complexity roughly sums up to $\mathcal{O}(|S|^2 \cdot |Act| + |S| \log |S| + \gamma(|Act|^2 + |S| \cdot |Act|))$. (The first part $\mathcal{O}(|S|^2 \cdot |Act| + |S| \log |S|)$ for the PGS itself can be considered optimal, while the second part depends on the efficiency of the LP algorithm being used.)

Time complexity. The number of variables in the LP problem in Lemma 7 is $|Act_{\mathtt{I}}| + |Act_{\mathtt{II}}|^2$, and the number of constraints is bounded by $1 + |Act_{\mathtt{I}}| + |Act_{\mathtt{II}}| +$

$|Act_{\mathrm{II}}|^2 + |S| \cdot |Act_{\mathrm{II}}|$. The predicate CanSim costs $|Act_{\mathrm{I}}|$ times LP solving. Each Split invokes at most $|B|^2$ testing of CanSim where B is a block in Σ. Each iteration of GCPP splits all current blocks, and the total number of comparisons within each iteration of GCPP is be bounded by $|S|^2$. (However it seems heuristics on the existing partition can achieve a speed close to linear in practice by caching previous CanSim checks [27].) The number of iterations is bounded by $|S|$. This gives us time complexity which is in the worst case to solve $\mathcal{O}(|Act_{\mathrm{I}}| \cdot |S|^3)$ many such LP problems, each of which has $\mathcal{O}(|S| \cdot |Act| + |Act|^2)$ constraints.

Remark. By removing the interaction between players (i.e., the alternating part), our algorithm downgrades to a partition-based algorithm computing the largest *strong* probabilistic simulation relation in probabilistic automata, where *combined transitions* are needed. The algorithm of [27] for computing strong probabilistic simulation has time complexity of solving $\mathcal{O}(|S|^2 \cdot m)$ LP problems, where m is the size of the transition relation comparable to $\mathcal{O}(|S|^2 \cdot |Act|)$. They have $\mathcal{O}(|S|^2)$ constraints for each LP instance. The improvement achieved in our algorithm is due to the use of partitions in each iteration instead of working on the whole relation, which is made possible by applying Lemma 3.

The space-efficient algorithm [26] for probabilistic simulation (*without* combined transitions) has the same space complexity but better time complexity than ours, which is due to the reduction to the maximal flow problem.

5 Conclusion

We have presented a partition-based algorithm to compute the largest probabilistic alternating simulation relation in finite probabilistic game structures. To the best of our knowledge, our work presents the first polynomial-time algorithm for computing a relation in probabilistic systems considering (concurrently) mixed choices from players. As aforementioned, PA-simulation is known as stronger than the simulation relation characterising quantitative μ-calculus [12], though it is still a conservative approximation which has a reasonable complexity to be useful in verification of game-based properties.

Acknowledgement. Wan Fokkink, Rob van Glabbeek and Lijun Zhang give us a lot of helpful comments. Especially we thank Timothy Bourke, whose comments have greatly improved the presentation of this paper.

References

1. R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time temporal logic. *Journal of ACM*, 49(5):672–713, 2002.
2. R. Alur, T. A. Henzinger, O. Kupferman, and M. Y. Vardi. Alternating refinement relations. In *Proc. CONCUR*, LNCS 1466, pages 163–178. Springer, 1998.
3. C. Baier, B. Engelen, and M. E. Majster-Cederbaum. Deciding bisimilarity and similarity for probabilistic processes. *Journal of Computer and System Sciences*, 60(1):187–231, 2000.

4. D. Bustan and O. Grumberg. Simulation based minimization. *ACM Transactions on Computational Logic*, 4(2):181–206, 2003.
5. S. Cattani and R. Segala. Decision algorithms for probabilistic bisimulations. In *Proc. CONCUR*, LNCS 2421, pages 371–386. Springer, 2002.
6. K. Chatterjee, L. de Alfaro, and T. A. Henzinger. The complexity of quantitative concurrent parity games. In *Proc. SODA*, pages 678–687. ACM, 2006.
7. K. Chatterjee, L. de Alfaro, R. Majumdar, and V. Raman. Algorithms for game metrics (full version). *Logical Methods in Computer Science*, 6(3:13):1–27, 2010.
8. E. M. Clarke and E. A. Emerson. Synthesis of synchronization skeletons for branching-time temporal logic. In *Proc. Workshop on Logics of Programs*, LNCS 131, pages 52–71. Springer, 1981.
9. L. de Alfaro. Quantitative verification and control via the mu-calculus. In *Proc. CONCUR*, LNCS 2761, pages 102–126. Springer, 2003.
10. L. de Alfaro, T. A. Henzinger, and O. Kupferman. Concurrent reachability games. In *Proc. FOCS*, pages 564–575. IEEE CS, 1998.
11. L. de Alfaro and R. Majumdar. Quantitative solution of omega-regular games. *Journal of Computer and System Sciences*, 68(2):374–397, 2004.
12. L. de Alfaro, R. Majumdar, V. Raman, and M. Stoelinga. Game refinement relations and metrics. *Logic Methods in Computer Science*, 4(3:7):1–28, 2008.
13. R. Gentilini, C. Piazza, and A. Policriti. From bisimulation to simulation: Coarsest partition problems. *Journal of Automatic Reasoning*, 31(1):73–103, 2003.
14. O. Grumberg and D. Long. Model checking and modular verification. *ACM Transactions on Programming Languages and Systems*, 16(3):843–871, 1994.
15. M. R. Henzinger, T. A. Henzinger, and P. W. Kopke. Computing simulations on finite and infinite graphs. In *Proc. FOCS*, pages 453–462. IEEE CS, 1995.
16. B. Jonsson and K. G. Larsen. Specification and refinement of probabilistic processes. In *Proc. LICS*, pages 266–277. IEEE CS, 1991.
17. N. Karmakar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4):373–395, 1984.
18. F. Ranzato and F. Tapparo. A new efficient simulation equivalence algorithm. In *Proc. LICS*, pages 171–180. IEEE CS, 2007.
19. R. Segala. *Modeling and Verification of Randomized Distributed Real-Time Systems*. PhD thesis, Massachusetts Institute of Technology, 1995.
20. R. Segala and N. A. Lynch. Probabilistic simulations for probabilistic processes. *Nordic Journal of Computing*, 2(2):250–273, 1995.
21. L. S. Shapley. Stochastic games. *Proc. National Academy of Science*, 39:1095–1100, 1953.
22. L. Tan and R. Cleaveland. Simulation revisited. In *Proc. TACAS*, LNCS 2031, pages 480–495. Springer, 2001.
23. R. J. van Glabbeek and B. Ploeger. Correcting a space-efficient simulation algorithm. In *Proc. CAV*, LNCS 5123, pages 517–529. Springer, 2008.
24. J. von Neumann and O. Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, 1947.
25. C. Zhang and J. Pang. On probabilistic alternating simulations. In *Proc. IFIP TCS*, AICT 323, pages 71–85. IFIP, 2010.
26. L. Zhang. A space-efficient probabilistic simulation algorithm. In *Proc. CONCUR*, LNCS 5201, pages 248–263. Springer, 2008.
27. L. Zhang, H. Hermanns, F. Eisenbrand, and D. N. Jansen. Flow faster: Efficient decision algorithms for probabilistic simulations. *Logical Methods in Computer Science*, 4(4:6):1–43, 2008.